ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

Heft 1 · 1990

Mikroprozessortechnik

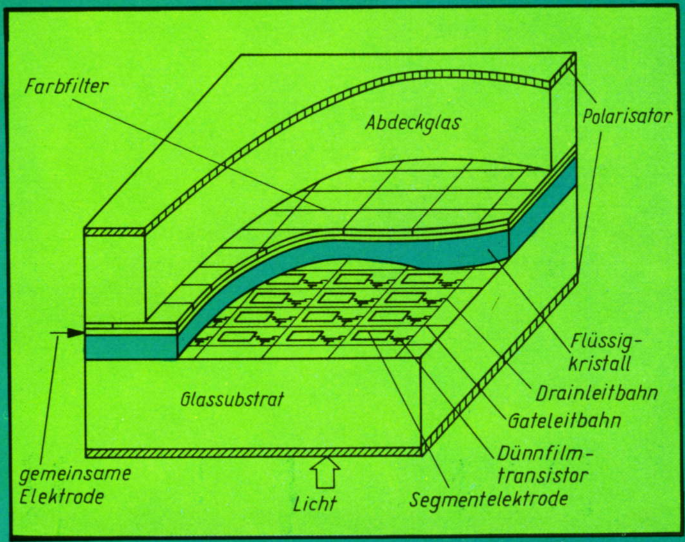
VEB Verlag Technik Berlin

ISSN 0232-2892

Zuverlässigkeit von Einchiprechner-Systemen

außerdem: ROM-Erweiterungen für IBM-PCs
Eingabemasken mit Turbo-Pascal
C-Compiler für CP/M im Test

Portable PCs mit Farbdisplays



Vor einem Jahr stellten wir Ihnen die Rechnerklasse Laptop vor – das sind die PCs, die man auf den Schoß (lap) nehmen kann. Seitdem konnten wir bei diesen Rechnern mehrere Entwicklungsrichtungen beobachten: Erstens hat sich die Zahl der Firmen, die Laptops oder transportable PCs (Portables) anbieten, stark erhöht; wir informierten Sie bereits über Neuentwicklungen der traditionellen Heimcomputerhersteller Schneider (in MP 7/89) und Atari (in MP 12/89). Zweitens wurden das Gewicht, der Stromverbrauch und der Preis verringert. Und drittens weisen die Geräte auch immer neue Leistungsmerkmale auf; neben der Integration von Festplattenlaufwerken und der Verwendung leistungsfähigerer und stromsparender Mikroprozessoren werden seit dem vorigen Jahr die ersten Portables mit Farbdisplays angeboten. In MP 6/1989 stellten wir kurz den Farb-Portable 386 von Sharp vor, der unter anderem einen 14-Zoll-Farb-LC-Bildschirm, einen 80386-Prozessor sowie eine 40-MByte-Festplatte besitzt.
Die Flachdisplays für Laptops bzw. Portables werden in verschiedenen Techniken hergestellt. Die meisten nichtfarbigen Displays sind Plasmadisplays – bestehend aus zwei Glasplatten (mit aufgebrachten Leitbahn-Elektroden), zwischen denen sich das aktiv leuchtende Medium befindet. Häufig werden auch Flüssigkristall-(LC)-Displays verwendet – bei denen sich zwischen zwei Glasplatten die Flüssigkristalle befinden, die nicht aktiv leuchten, sondern, abhängig von einem elektrischen Feld, eine unterschiedliche Lichtdurchlässigkeit aufweisen (hier gibt es inzwischen eine Vielzahl von LCD-Typen). Einige wenige Hersteller verwenden Elektrolumineszenz-(EL-)Displays, die aus nur einer Glasplatte aufgebaut sein können, auf der abwechselnd Leitbahn-Elektroden, Isolationsschichten und eine fluoreszierende, aktiv leuchtende Schicht aufgebracht wurden. Vergleicht man die Zahlen der in Entwicklung befindlichen Flach-Farbdisplays für Portables, kann man feststellen, daß die LC-Displays vor den Plasmadisplays und den EL-Displays rangieren. Die zur Zeit angebotenen Farbdisplays für Laptops bzw. Portables sind ausschließlich LC-Displays. Beispielsweise entwickelte die japanische Firma Hitachi für den Farb-Laptop HL 400 C – nach eigenen Angaben der weltweit erste Laptop mit Farbdisplays – ein Dünnfilmtransistor-LC-Display. Dieses Display mit einer Diagonalen von 6,3 Zoll (1 Zoll = 2,54 cm) enthält 640 × 200 (× 3) Bildpunkte (CGA-kompatibel). Im Textmodus können 25 Zeilen mit je 70 oder 80 Zeichen dargestellt werden. Die verwendete Dünnfilmtransistor-Technik (auf der Basis von amorphem Silizium) soll gegenüber den herkömmlichen Super-Twisted-LCDs eine zehnfach schnellere Reaktionszeit und einen besseren Kontrast bieten. Mit dem Blickwinkel von <50° (von der z-Achse aus) wird ebenfalls Super-Twisted-Qualität erreicht. Durch die Reaktionszeit von 50 ms sind bei Grafikanwendungen Bewegungen der Maus praktisch ohne Verzögerung möglich. Jeder Farbpunkt wird von einem Transistor angesteuert (siehe Skizze), der zusätzlich als Dünnfilm auf der Glasoberfläche abgeschieden wurde. Drei Farbpunkte bilden einen Bildpunkt; die Farbgebung wird durch einen Filter erreicht. Durch RGB-Ansteuerung lassen sich so acht Farben (gleichzeitig) darstellen. Die Punktgröße beträgt 0,16 × 0,2 mm^2. Eine Hintergrundbeleuchtung ist möglich. Die Leistungsaufnahme des Displays kommt auf stattliche 1,5 Watt; daraus resultiert sicherlich die geringe Akku-Betriebsdauer von nur einer Stunde. Darüber hinaus bietet der HL 400 C (C steht für Color) alles, was man von einem voll AT-kompatiblen Laptop erwarten kann: eine CPU 80C286 in stromsparender CMOS-Technik mit 12,5-MHz-Taktrate, einen 1-MByte-Arbeitsspeicher (auf Kosten eines Steckplatzes auf 2 MByte aufrüstbar), eine 3,5-Zoll-Festplatte mit 20 MByte und das 3,5-Zoll-Floppy-Laufwerk. Der HL 400 C enthält eine parallele, zwei serielle und eine RGB-Schnittstelle. Dabei sind immer noch zwei Steckplätze für Erweiterungskarten verfügbar.
Auch die japanische Firma Toshiba – weltgrößter Produzent von Speicherschaltkreisen – kündigte inzwischen einen Laptop mit Farb-LC-Display an. Er soll bereits VGA-kompatibel – mit 640 × 480 Punkten und bis zu 16 Farben – sein; die Bildschirmdiagonale mit 11 Zoll ist fast doppelt so groß wie die des HL 400 C-Displays. Während dieser Laptop 1990 in Serie gehen soll, befindet sich auf dem Reißbrett bereits ein 14-Zoll-Display mit 720 × 500 Bildpunkten, das eine Gemeinschaftsentwicklung mit IBM darstellt (vergleiche auch MP 11/1989, S. 349).

MP-Hk

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
4. Jahrgang · ISSN 0232–2892

Mikroprozessortechnik, Heft 1 · 90

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR – 1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Hans Weiß, Verantwortlicher Redakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2870203), Hans-Joachim Hill (Tel. 2870209); Sekretariat Tel. 2870381

Gestaltung Christina Bauer

Titelfoto Thomas Bölkow

Beirat Dr. Ludwig Claßen, Dr. Heinz Florin, Prof. Dr. sc. Rolf Giesecke, Joachim Hahne, Prof. Dr. sc. Dieter Hammer, Prof. Dr. sc. Thomas Horn, Prof. Dr. Albert Jugel, Prof. Dr. Bernd Junghans, Dr. Dietmar Keller, Prof. Dr. sc. Gernot Meyer, Prof. Dr. sc. Bernd-Georg Münzer, Prof. Dr. sc. Peter Neubert, Prof. Dr. sc. Rudolf Arthur Pose, Prof. Dr. sc. Dr. Michael Roth (Vorsitzender), Dr. Gerhard Schulze, Prof. Dr. sc. Manfred Seifart, Dr. Dieter Simon, Dr. Rolf Wätzig, Prof. Dr. sc. Dr. Jürgen Zaremba

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

Erfüllungsort und Gerichtsstand Berlin-Mitte. Der Verlag behält sich alle Rechte an den von ihm veröffentlichten Aufsätzen und Abbildungen, auch das der Übersetzung in fremde Sprachen, vor. Auszüge, Referate und Besprechungen sind nur mit voller Quellenangabe zulässig.

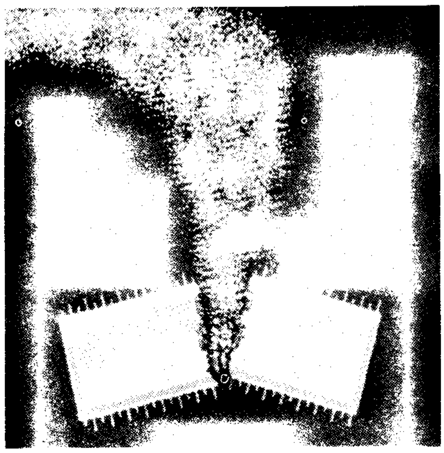
Redaktionsschluß 8. November 1989

AN (EDV) 49837

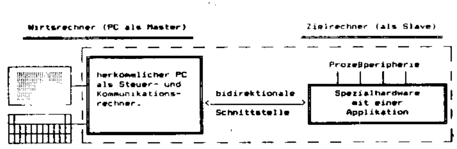
Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propagandititet Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Republik Ungarn:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nôi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR – 7010 Leipzig und Leipzig Book Service, Talstraße 29, DDR – 7010 Leipzig



Systeme mit Einchipmikrorechnern – im angloamerikanischen Sprachgebrauch auch Mikrocontroller genannt – weisen gegenüber herkömmlichen Mikrorechnersystemen eine höhere Zuverlässigkeit auf. Vorausgesetzt, die Einchipmikrorechner werden richtig eingesetzt, ansonsten kann es einem Schaltungsentwickler schon einmal so ergehen, wie auf unserem Titelbild dargestellt.
In unserem Beitrag „Zuverlässigkeit von Einchipmikrorechnersystemen" auf **Seite 3** werden nach der Behandlung einiger Begriffe der Zuverlässigkeitstheorie Maßnahmen zur Zuverlässigkeitserhöhung abgeleitet und deren Anwendung beim Einsatz von Einchipmikrorechnern erläutert.



Das Entwickeln, Testen und Inbetriebnehmen von Software für Rechner ohne eigene Standardperipherie (Bildschirm und Tastatur) oder auch für Einchipmikrorechner ist oft nur mit Crossassemblern und Emulatoren möglich.
Unter dem Titel „Ein verteiltes Programmentwicklungskonzept für Mikrorechner" stellen wir Ihnen auf der **Seite 23** ein Verfahren vor, mit dem – beispielsweise von einem PC aus – ein Zielrechner sowohl interaktiv programmiert als auch beobachtet und gesteuert werden kann.

Vorschau

Für das Heft 2/1990 bereiten wir für Sie unter anderem Beiträge zu folgenden Themen vor:
- Mikroelektronik und Meßwerterfassung
- Bussysteme für Multiprozessoranordnungen
- Funktionsinterpreter in Turbo-Pascal

Inhalt

Technik international — **2. US**
Portable-PCs mit Farbdisplays

MP-Info — **2**

Carsten Knop:
Zuverlässigkeit von Einchipmikrorechnersystemen — **3**

Bernd Matzke:
Eingabemasken mit Turbo-Pascal — **5**

Jens Immig:
ROM-Erweiterungen für IBM-PCs — **8**

Uwe Schulze:
Im Test: C-Compiler für CP/M — **9**

Kathrin Lajow, Wolfram Kammer, Wolfgang Spindler:
Farbgrafikadapter für K 1520-Systeme – klein und vielseitig — **11**

Jahresinhaltsverzeichnis 1989 — **15**

Christian Hanisch:
Arrangiert: Turbo-Pascal und ANSI — **21**

Uwe Wloka:
Datenbankbetriebssystem MIMER auch auf dem PC — **22**

Ralf Neuthe:
Ein verteiltes Programmentwicklungskonzept für Mikrorechner — **23**

MP-Computer-Club — **26**

Dimiter Mantschew:
MS-DOS-Tip: Tastaturbelegung ändern
Bernd Voitel:
Temperaturmessung mit dem KC 85/1
Wolfram Schütze:
Nutzung der Fenstervektoren für Basic
Thomas Fleischer:
Gelöschte Basic-Programme erreichen
Sylvia Springer, Knut Glockmann:
Textverarbeitung von Datenbanken

MP-Börse — **28**

Entwicklungen und Tendenzen — **30**

vorgestellt — **32**
Industriestandard-PCs von DEC

vorgestellt — **4. US**
Atari PC 4

Patentrecherchen per Fernzugriff

In der Erzeugnis- und Verfahrensentwicklung sowie in der Produktion gilt es, mit wissenschaftlich-technischen Spitzenleistungen weiter an Tempo zuzulegen. Das erfordert in den Kombinaten und Betrieben, den aktuellen internationalen Stand auf dem jeweiligen speziellen Gebiet genau zu kennen, die Entwicklungsrichtungen von Wissenschaft und Technik treffsicher zu bestimmen, die einen stabilen Effektivitäts- und Qualitätszuwachs in der Produktion bewirken sollen. Aufgabe der Generaldirektoren der Kombinate ist es, mit den Pflichtenheften die Aufgaben und hohen Leistungsansprüche für neue Erzeugnisse und Verfahren exakt festzulegen.
Eine wichtige Informationsquelle für solide Kenntnis des wissenschaftlich-technischen Weltstandes von Erzeugnissen, Technologien und Verfahren stellt die Patentliteratur dar. Die veröffentlichten Patentdokumente des In- und Auslandes enthalten die umfassendste Sammlung über die in der Welt bereits bekannten technischen und technologischen Ideen und Lösungen. Deren gründliches Studium und wissenschaftliche Analyse sind für eigene anspruchsvolle Aufgabenstellungen unerläßlich, will man den internationalen Anforderungen gerecht werden und den Weltstand mitbestimmen.
Das Amt für Erfindungs- und Patentwesen bietet Kombinaten und Einrichtungen vielfältige Informationsdienste über weltweit veröffentlichte Patente an, so beispielsweise statistische und bibliographische Informationen, Kurzfassungen und vollständige Texte zu Erfindungsbeschreibungen. Vor allem die vom Amt bereitgestellten Patentdatenbanken erweisen sich als besonders effektiver Service. Seit 1988 haben Interessenten in unserer Volkswirtschaft die Möglichkeit, die Datenbank „DDR-Patentdokumente" und „Neuveröffentlichte Erfindungsbeschreibungen des Welt-Patentfonds" im Fernzugriff per Computer und Bildschirm zu nutzen.
1989 wurde das Informationsangebot um die thematische Patentdatenbank „Mikroelektronik" erweitert. Diese Informationsfonds stellt das Rechenzentrum für Patentdatenbanken des VEB Datenverarbeitungszentrum Frankfurt/Oder bereit.
Gegenwärtig nutzen 50 Kombinate sowie Forschungs- und Entwicklungseinrichtungen diese Datenbanken auf der Grundlage von Vereinbarungen mit unserem Amt. In Abstimmung mit dem Amt für Erfindungs- und Patentwesen können sich weitere Interessenten den erwünschten Fernzugriff per Bildschirm über das Datennetz der Deutschen Post erschließen. Zunehmend mehr Einrichtungen können so direkt vom Bildschirmarbeitsplatz aus in der Patentliteratur „nachschlagen".
Das Amt für Erfindungs- und Patentwesen arbeitet gegenwärtig zielstrebig daran, die Verfügbarkeit der vorhandenen Datenbanken zu erhöhen und durch weitere Patentdatenbanken das Informationsangebot zu verbreitern.

L. Janke

Hinweise für unsere Autoren

1. Die einseitig mit Maschine beschriebenen Manuskripte – **2zeilig, 60 Anschläge/Zeile, 30 Zeilen/ Seite** senden Sie bitte **in zweifacher Ausfertigung** an den **VEB Verlag Technik, Redaktion MP, Oranienburger Straße 13/14, Berlin, 1020.**
Als Manuskript können auch **Computerausdrucke** mit einem sauberen und kontrastreichen Schriftbild dienen, sofern durchschnittlich 60 Zeichen pro Zeile eingehalten werden (also wie bei Schreibmaschine **mit Silbentrennung und ohne Blocksatz!**). Bei Verwendung von Matrixdruckern drucken Sie bitte mindestens in **Near Letter Quality (NLQ)**.
2. Der Umfang des Manuskriptes soll im allgemeinen 10 Seiten nicht überschreiten. Die maximale Länge des Textteils von Angeboten und Suchmeldungen **für die Rubrik „Börse"** beträgt **30 Zeilen**.
3. Das Manuskript beginnt mit einer knappen, aber aussagekräftigen Überschrift; es folgen der akademische Grad, der volle Vor- und Zuname und der Ort.
4. Versuchen Sie, dem Leser die Aufnahme der Informationen zu erleichtern, indem Sie bei der Gestaltung des Textteiles folgendes beachten:
- Der Beitrag muß übersichtlich gegliedert sein! (Zwischenüberschriften, Absätze; eventuell kurze Zusammenfassungen oder die Hervorhebung von Kernsätzen)
- Wo es der Aussage oder der Anschaulichkeit dient, verwenden Sie bitte Bilder, Tafeln oder Fotos (siehe auch Punkt 6).
- Versuchen Sie bitte, Ihren Beitrag in einem leichtverständlichen Schreibstil abzufassen.

5. Verwenden Sie die Abkürzungen, die im Duden aufgeführt sind; zusätzlich abzukürzende Begriffe schreiben Sie bitte beim ersten Auftreten aus und setzen die Abkürzung in Klammern.
6. Legen Sie bitte **Bilder, Tafeln und Fotos** als Anhang getrennt numeriert bei und kennzeichnen die Anordnung im Text. Die erläuternden Unterschriften für **alle Bilder und Fotos** schreiben Sie bitte auf ein gesondertes Manuskriptblatt mit dem Vermerk „Bildunterschriften".
Bilder und Tafeln sollten übersichtlich und eindeutig lesbar maschinegeschrieben (auch von Drucker oder Plotter) oder als Bleistiftzeichnung eingereicht werden. Um Satzfehler, insbesondere bei **Programmlistings** zu vermeiden, liefern Sie diese bitte **einzeilig** als **reproduktionsfähige Vorlage** (kontrastreicher Druck mit schwarzem Farbband in NLQ oder Schreibmaschinenschrift auf weißem Papier).
Bitte beachten Sie, daß auch die eingesandten Programme frei von den Rechten Dritter sein müssen.
Schwarzweißfotos sollten das Format 13 cm × 18 cm haben; das Negativ benötigen wir nicht.
Farbfotos stimmen Sie bitte vorher mit der Redaktion ab. Geben Sie bei honorarpflichtigen Fotos den Namen und die Adresse bzw. die Konto-Nummer des Bildautors an!
7. Bedenken Sie bitte, daß die Anzahl der Zeichen pro Zeile im Druck geringer ist als im Manuskript. Passen Sie also im Interesse einer korrekten Wiedergabe die Gleichungen und Formeln, kurze Programmauszüge u. ä., die im laufenden Text erscheinen sollen, von vornherein an die **Spaltenbreite** (etwa 40 Zeichen) an. Wo dieses nicht möglich ist, deklarieren Sie diese Zeilen als Bilder.
8. **Literaturverweise** werden, in Schrägstrichen eingeschlossen, **fortlaufend numeriert**, z. B. „wie in /1/ und /2/ dargestellt wurde", oder „auch Meyer hat in /3/ darauf hingewiesen". Das Verzeichnis der verwendeten Literatur fertigen Sie bitte auf einem gesonderten Manuskriptblatt in der Reihenfolge des Bezugs im Text nach folgendem Schema an:

Literatur
/1/ Menzer, R.; Richter, B.: Neue Technologien für Digitalisiergeräte. Feingerätetechnik, Berlin 34 (1985) 9, S. 386
/2/ Claßen, L.; Oefler, U.: Wissensspeicher Mikrorechnerprogrammierung. VEB Verlag Technik, Berlin 1986

9. Zu jedem Fachartikel schreiben Sie bitte ein etwa 10 Zeilen umfassendes **Referat**, worin das Wesentliche des Beitrages enthalten ist.
10. Bei längeren Fachaufsätzen bitten wir Sie darüber hinaus um ein kurzes **Autorporträt** mit Angaben zu Alter, Ausbildung, beruflichem Werdegang, jetziger Tätigkeit und Aufgabenschwerpunkten.
11. Unterstützen Sie bitte unsere Bemühungen, den Erfahrungsaustausch zu fördern, indem Sie für Interessenten eine betriebliche Kontaktadresse und eine Telefonnummer zur Veröffentlichung angeben.
12. Denken Sie bitte auch daran, daß bei Fachbeiträgen aus Ihrem Arbeitsbereich eine Veröffentlichungsgenehmigung Ihrer Dienststelle erforderlich ist.
13. Nicht zu vergessen sind schließlich die private und die dienstliche Anschrift, Telefonnummern für eventuelle Rückfragen sowie Ihre Konto-Nummer.
Bitte überprüfen Sie das Manuskript vor dem Absenden noch einmal anhand der gegebenen Hinweise – Sie vermeiden unnötige Verzögerungen bei der Bearbeitung und der Veröffentlichung Ihres Beitrages.

Ihre Redaktion MP

Mikroelektronik: Großer Nutzen auch in kleinen Betrieben

Charakteristisch für den Industriebereich der bezirksgeleiteten und Lebensmittelindustrie sind eine Reihe großer zentralgeleiteter Kombinate, aber auch eine Vielzahl kleinerer Betriebe insbesondere der bezirksgeleiteten Industrie. Ihr Produktionsprofil ist breit: Neben den in allen Bezirken ansässigen und für die Bevölkerungsversorgung mit Waren des täglichen Bedarfs bedeutsamen Getränke- und Backwarenproduzenten zählen z. B. leistungsstarke Klein- und Mittelbetriebe der metallverarbeitenden, chemischen, Leicht- und der Schnittholzindustrie sowie Rationalisierungsmittelbaubetriebe dazu.
Eine Reihe von ihnen verfügt über gute Erfahrungen bei der Anwendung der Mikroelektronik, weiß die Vorteile dieser Schlüsseltechnologie zu schätzen, ob bei der rationelleren Gestaltung des Reproduktionsprozesses, in der Erzeugnisentwicklung, hinsichtlich einer material- und energieökonomischen oder zeitsparenden Fertigung sowie verbesserten Arbeitsbedingungen. Derzeit sind in Betrieben der bezirksgeleiteten Industrie z. B. rund 1 200 CAD/CAM-Stationen im Einsatz. Zugleich: Auch Kleinbetriebe lassen zunehmend die Mikroelektronik in ihre Erzeugnisse einfließen. Das Gütezeichen „Q", Patentanmeldungen, Goldmedaillen zu Leipziger Messen und Prädikate für Spitzenleistungen sind Ausdruck für Fortschritte in der Arbeit auf diesem Gebiet.
Die Möglichkeiten der Mikroelektronik machten sich z. B. die Werktätigen im bezirksgeleiteten VEB Berliner Strickmoden zunutze. Rechnergestützt erfolgt hier der Strickmusterentwurf, einschließlich der Farbmusterung, sowie die damit sofort mögliche Steuerung der Fachstrickautomaten. Der Nutzen liegt auf der Hand: Das Strickwarenangebot wird vielseitiger, die Erneuerungsrate erhöht sich auf 80 Prozent, der Zeitaufwand für die Musterung sinkt um 95 Prozent, spürbar verkürzt wird ebenso der Zeitraum vom Musterentwurf bis zur Auslieferung der fertigen Erzeugnisse. Im VEB Metallwerkstätten „Warnow" Rostock konnte mit Hilfe der Mikroelektronik die Produktion von Schäkeln – Anschlagmitteln für Hebezeuge – automatisiert werden. Die Ablaufsteuerung ist effektiver gestaltet und das Prüfen automatisiert worden, so daß die Arbeitsproduktivität um 117 Prozent stieg und 6 Werktätige andere Aufgaben übernehmen konnten.
Eingang findet die Mikroelektronik in einer Vielzahl von Erzeugnissen, die in Klein- und Mittelbetrieben des Bereiches hergestellt werden.
Deutlich zeigt sich also: Die gezielte Anwendung der Schlüsseltechnologie Mikroelektronik bringt auch in kleinen Betrieben unmittelbaren Leistungszuwachs, trägt zugleich bei, das Angebot an hochwertigen Konsumgütern zu bereichern und der Forderung nach anspruchsvollen Zulieferungen gerecht zu werden.

Prof. Dr. K. Borkmann

Zuverlässigkeit von Einchip-mikrorechnersystemen

Carsten Knop
VEB Forschung, Entwicklung und Rationalisierung des Schwermaschinen- und Anlagenbaus Magdeburg

[illegible]sten Kn[illegible] 1977 bis 1982 an der Technischen Hochschule Ilmenau, Sektion Technische und Biomedizinische Kybernetik. Von 1982 bis 1984 war er an der AdW der DDR tätig. Seit 1984 arbeitet er im VEB Forschung, Entwicklung und Rationalisierung des Schwermaschinen- und Anlagenbaus auf dem Gebiet der Mikrorechentechnik und Digitaltechnik zur Prozeßautomatisierung. 1987 nahm er eine außerplanmäßige Aspirantur an der Sektion Elektronik der Hum[illegible] Univers[illegible] zu Berlin auf.

Für die Lösung einer Vielzahl von Aufgaben in der Kleinautomatisierung stellt die Familie der Einchipmikrorechner (EMR) ein leistungsfähiges und kostengünstiges Instrumentarium dar. Vor allem in Geräten, Baugruppen und kleineren elektronischen Funktionseinheiten werden EMR zur Überwachung von Prozessen und zur Steuerung von Funktionsabläufen verwendet. Die Konfigurierbarkeit der Ports, die freie Wahl des Bearbeitungsalgorithmus durch das Programm und die möglichen Peripheriefunktionen sorgen für die Flexibilität und damit für das breite Anwendungsspektrum der EMR. Gleichzeitig ergeben sich aber auch Möglichkeiten zur Beeinflussung der Zuverlässigkeit des jeweiligen technischen Systems, die viele Entwickler heute noch nicht konsequent nutzen. Im folgenden werden deshalb einige Begriffe der Zuverlässigkeitstheorie behandelt, allgemeine Maßnahmen zur Zuverlässigkeitserhöhung erläutert und deren praktische Anwendung beim Einsatz von EMR abgeleitet.

Begriffe

Ganz allgemein ist nach /1/ die Zuverlässigkeit ein Ausdruck für das Vorhandensein von Gebrauchseigenschaften innerhalb einer bestimmten Nutzungszeit. Durch das Wirken von äußeren und inneren Einflüssen können sich diese Gebrauchseigenschaften verändern. Dabei kann es zu einem Fehler oder zu einem Ausfall des jeweiligen Systems kommen. /2/ charakterisiert diese Begriffe folgendermaßen:

– Ein Fehler liegt vor, wenn die volle Funktionsfähigkeit eines Systems nicht mehr gewährleistet ist, das heißt, es werden nicht mehr alle Haupt- und Nebenparameter erfüllt (z. B. defekte Kontrollampe in einem Stromversorgungsgerät).

– Ein Ausfall liegt dann vor, wenn die Arbeitsfähigkeit eines Systems nicht mehr gewährleistet ist, das heißt, es werden nicht mehr alle Hauptparameter erfüllt (z. B. Kurzschluß in einem Stromversorgungsgerät).

Zur Beurteilung der Zuverlässigkeit von Systemen werden die Kenngrößen nach /3/ herangezogen. Ihre quantitative Ermittlung erfolgt mit statistischen und wahrscheinlichkeitstheoretischen Mitteln. Bei der praktischen Anwendung ist aber oftmals eine vergleichende Beurteilung mehrerer möglicher Varianten sinnvoll und ausreichend.

Für allgemeine Aussagen wird oft die Dauerverfügbarkeit V_D verwendet. Hierbei kommen mehrere Zuverlässigkeitskenngrößen gleichzeitig zur Anwendung:

$$V_D = \frac{\Theta}{\Theta + \overline{T_A}} \qquad (1)$$

mit: Θ = mittlerer Ausfallabstand
$\overline{T_A}$ = mittlere Ausfalldauer

Diese komplexe Kenngröße faßt die Erwartungen eines Anwenders in bezug auf die Zuverlässigkeit eines technischen Systems zusammen:

– Die Arbeitsfähigkeit muß über einen möglichst langen Zeitraum erhalten bleiben.

– Wenn es zu einem Ausfall kommt, muß dieser in möglichst kurzer Zeit behebbar sein.

Maßnahmen zur Zuverlässigkeitserhöhung

Ausgangspunkt für die Entwicklung eines technischen Systems ist die gründliche Analyse des Prozesses, in dem das System eingesetzt werden soll. Ergibt sich daraus die Forderung nach erhöhter Zuverlässigkeit und Dauerverfügbarkeit, kommen die Prinzipien der Fehlerdiagnose und der Fehlertoleranz zum Einsatz. Die Fehlerdiagnose realisiert eine Überprüfung des Einschaltzustandes und eine zyklische Eigenüberwachung des Systems, ohne den Funktionsumfang im fehlerfreien Fall wesentlich einzuschränken.

Im Fehlerfall schließt sich an die Fehlererkennung die Fehlerlokalisation an. Die Diagnosetiefe ist dabei von der weiteren Auswertung des Fehlersignals abhängig. In einfacheren Systemen wird zumeist nur der Fehler angezeigt und der Bediener damit zu einer Reaktion aufgefordert (z. B. Papierendeerkennung an einem Drucker). Das Ergebnis der Fehlerdiagnose wird aber auch dazu genutzt, das System in einen für den angeschlossenen Prozeß sicheren Zustand zu überführen (Fail-save-Verhalten). Beispiele hierfür sind das Abschalten eines Antriebes beim Ausfall eines Wegmeßsystems, um eine Kollision zu vermeiden, oder das Einschalten eines Lüfters beim Ausfall der Temperaturmessung in einer Steuerung, um eine Überhitzung zu verhindern. Weiterhin kommen die Prinzipien der Fehlerdiagnose bei der Instandsetzung zur Anwendung. Eine hinreichend exakte Fehlerlokalisation (z. B. Ermittlung des defekten Moduls) verkürzt die mittlere Ausfalldauer $\overline{T_A}$ und wirkt nach (1) positiv auf die Dauerverfügbarkeit V_D. Voraussetzung für eine Fehlertoleranz ist das Vorhandensein redundanter Systemelemente. Fehlertoleranz wird in folgenden Schritten erreicht (vgl. /4/ und /5/):

- Fehlerdiagnose mit Fehlererkennung und -lokalisation
- Rekonfiguration
- Wiederanlauf.

Damit wird der mittlere Ausfallabstand Θ gegenüber Systemen ohne Fehlertoleranz vergrößert, was sich nach (1) positiv auf die Dauerverfügbarkeit V_D auswirkt.

Zuverlässigkeit in EMR-Schaltungen

Im folgenden werden einige allgemeine Beispiele für Zuverlässigkeitsmaßnahmen erläutert. Dabei wird der Schwerpunkt auf die Frage der Fehlerdiagnose gelegt. Speziell auf dem Gebiet der Kleinautomatisierung stellt sie ein wichtiges Instrumentarium der Zuverlässigkeitsarbeit dar. In diesem Beitrag werden nur permanente und transiente Hardwarefehler betrachtet. Da es sich bei Softwarefehlern ausschließlich um Entwurfs- und Programmierfehler handelt, können sie in kleinen Systemen, die ausreichend getestet sind, genauso vernachlässigt werden wie Hardwareentwurfsfehler. Ein allgemeiner Programmablauf eines EMR-Systems ist in Bild 1 dargestellt. Er verdeutlicht den sequentiellen Zusammenhang der einzelnen Diagnoseabschnitte.

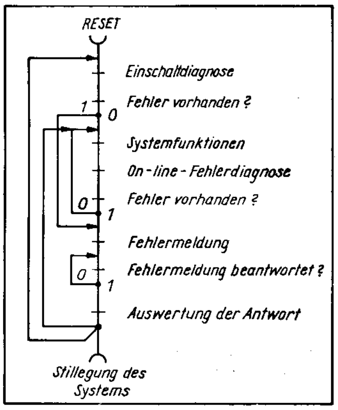


***Bild 1** Allgemeiner Programmablauf eines EMR-Systems*

Nach dem Rücksetzen des EMRs wird zunächst eine Einschaltdiagnose durchlaufen. Im allgemeinen besteht an sie keine Echtzeitanforderung, da die Systemfunktionen zu diesem Zeitpunkt noch nicht arbeiten. Die Einschaltdiagnose wird in mehreren Stufen durchgeführt. Als erstes werden Funktionen überprüft, die unmittelbar den EMR betreffen. Anschließend erfolgt die Überprüfung der Hardwareumgebung des EMRs und zum Schluß die Testung der vorhandenen Prozeßschnittstellen. Dadurch wird weitgehend gewährleistet, daß jeder neue Test nur auf bereits überprüften Systemkomponenten aufbaut.

Zu Beginn der Einschaltdiagnose findet ein ROM-Test statt, da das Vorhandensein des fehlerfreien Speicherinhalts eine Hauptvoraussetzung für die ordnungsgemäße Programmabarbeitung ist. Dieser Test wird meist als Prüfsummenbildung oder CRC-Berechnung über den gesamten ROM durchgeführt. Der Vergleich des ermittelten Ergebnisses mit dem als Konstante gespeicherten Sollergebnis liefert die Entscheidung, ob ein Fehler vorliegt. Anschließend werden die allgemein verwendbaren Register einem RAM-Test unterzogen. Bild 2 zeigt dazu ein Programmbeispiel zum Überprüfen der Schreib- und Lesefunktionen. In ähnlicher Weise ist auch der eventuell vorhandene externe RAM zu prüfen. Weitere EMR-Funktionen sind auf Grund der Komplexität des Bauelementes in der Regel nicht sinnvoll und umfassend zu testen. Ihre Fehlerfreiheit kann nur indirekt an der ordnungsgemäßen Programmabarbeitung überprüft werden. Eine Möglichkeit dazu ist die Installation eines Watch-dog-Ti-

```
TEST:   CLR %4          ;Test des Registers R5
MA1:    LD %5,%4
        CP %4,%5
        JP NZ,FEHL
        INC %4
        JR NZ,MA1
        CLR %5          ;Test des Registers R4
MA2:    LD %4,%5
        CP %5,%4
        JP NZ,FEHL
        INC %5
        JR NZ,MA2
        LD %4,#%6       ;Test der Register R6 - R127
MA3:    CLR %5
MA4:    LD @%4,%5
        CP %5,@%4
        JP NZ,FEHL
        INC %5
        JR NZ,MA4
        INC %4
        CP %4,#%8Ø
        JR NZ,MA3
        .
        .
FEHL:   ....            ;Fehlerbehandlung
```

Bild 2 Programmteil zum Testen der Register

mers nach /6/, der vom EMR zyklisch angesprochen wird. Bleibt dieses Ansprechen aus, so ist auf einen Fehler in der Arbeitsweise des EMRs zu schließen. Im zweiten Teil der Eigendiagnose erfolgt die Überprüfung der Hardwareumgebung des EMR. Bild 3 zeigt dazu eine Möglichkeit der prinzipiellen Herangehensweise. Mit dieser Anordnung können reale Prozeßdaten durch den EMR simuliert und die ordnungsgemäße Arbeitsweise der Vorverarbeitungshardware überprüft werden.

Das zusätzliche Element Multiplexer (MUX) stellt die Voraussetzung für die Fehlerdiagnose in der Vorverarbeitungshardware dar und wirkt dadurch positiv auf die mittlere Ausfalldauer $\overline{T_A}$. Außerdem bildet es die Grundlage für die Realisierung eines Fail-save-Verhaltens und für die Anwendung von Elementen der Fehlertoleranz. Bei richtiger Simulation der Prozeßsignale kann mit dieser Anordnung ein vorhandener Fehler mit hoher Wahrscheinlichkeit erkannt werden.

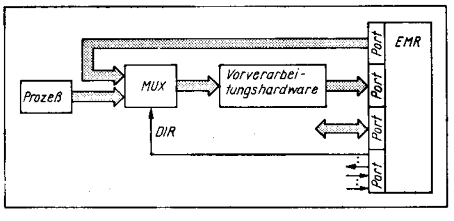


Bild 3 Prinzipschaltung zum Hardwaretest

Die Funktionsweise der Prozeßschnittstelle kann in der Regel nicht vollständig getestet werden. Mit einigen einfachen Maßnahmen lassen sich aber bestimmte Fehler erkennen. Dazu sind die Meßglieder des EMR-Systems abzufragen und die ermittelten Prozeßdaten auf die Einhaltung eines vorgegebenen Wertebereiches zu testen. Liegt eine Bereichsüberschreitung vor, so ist auf einen Fehler in der Prozeßschnittstelle oder im Prozeß zu schließen. Wenn der Prozeß es erlaubt, kann an dieser Stelle noch eine gezielte Prozeßbeeinflussung über die Stellglieder erfolgen. Die entsprechende Reaktion der Meßglieder dient auch hier der Fehlererkennung.

Beim Auftreten eines Fehlers generiert der EMR eine Fehlermeldung. Sie wird in Abhängigkeit von der Systemkonfiguration dem Bediener angezeigt, zum übergeordneten Rechner übertragen oder für die Aktivierung von Fehlertoleranzfunktionen genutzt.

Eine Reaktion auf die Fehlermeldung wertet der EMR aus, und in Abhängigkeit davon erfolgt eine Stillegung (evtl. mit Fail-save-Verhalten), ein Neustart oder eine Fortsetzung der Arbeit des Systems.

Nach dem fehlerfreien Ablauf der Eigendiagnose geht das System zu seiner normalen Funktion über (s. Bild 1). Im Hintergrund wird dabei die Online-Fehlerdiagnose durchgeführt. Sie überwacht zyklisch das System und kann immer aktiviert sein, wenn der EMR nicht für die Systemfunktionen benötigt wird. Es kommt ein großer Teil der Tests zur Anwendung, die bei der Einschaltdiagnose bereits erläutert wurden. Auch weitere Testfunktionen, wie die Überwachung der Versorgungsspannung nach /6/, können sinnvoll angewendet werden. Da nicht bei jedem Aufruf der Online-Fehlerdiagnose alle Tests durchgeführt werden können, ist darauf zu achten, daß Elemente mit hoher Ausfallwahrscheinlichkeit öfter überprüft werden als andere Elemente.

Bei kleineren EMR-Systemen finden die Prinzipien der Fehlertoleranz nur begrenzt Anwendung. Hauptgrund dafür ist der erhöhte Aufwand zur Schaffung redundanter Systemfunktionen. Einen Kompromiß zwischen Aufwand und Leistungsfähigkeit nach dem Auftreten eines Fehlers stellt die Redundanzart Graceful Degradation dar. Diese in /4/ ausführlich beschriebene Redundanzart geht von einer definierten Leistungsminderung im Fehlerfall aus, wobei grundlegende Systemfunktionen aufrechterhalten werden.

Eine weitere zur Anwendung kommende Methode ist die Zeitredundanz. Dabei werden bestimmte Programmteile mehrfach ausgeführt und die Ergebnisse verglichen. Es läßt sich damit ohne zusätzlichen Hardwareaufwand eine Toleranz gegenüber transienten Fehlern erreichen. Die erhöhte Programmlaufzeit ist für viele kleinere EMR-Systeme unkritisch, da keine harten Echtzeitanforderungen bestehen.

Ausblick

Die Bedeutung der Zuverlässigkeit als Bestandteil der Erzeugnisqualität wird auch in den nächsten Jahren stark zunehmen. Diesen Trend unterstützen heute schon einige Bauelementehersteller durch die Bereitstellung von Diagnose- und Fehlertoleranzprinzipien als Hard- und Firmware (vgl. /7/ und /8/). Zur Durchsetzung von Zuverlässigkeitsmaßnahmen in der Systemebene wird der Entwickler aber in immer stärkerem Maße die Prinzipien der Fehlerdiagnose und der Fehlertoleranz zur Anwendung bringen müssen.

Literatur

/1/ Nikolaizik, J.: VEM-Handbuch Zuverlässigkeit von Automatisierungs- und Elektroenergieanlagen. Berlin: VEB Verlag Technik 1981
/2/ TGL 26096/01: Zuverlässigkeit in der Technik, Begriffe.
/3/ TGL 26096/03: Zuverlässigkeit in der Technik, Auswahl von Zuverlässigkeitskenngrößen.
/4/ Kriesel, W.; Schäfer, M.: Fehlertolerante Mikrorechner-Funktionseinheiten mit der Redundanzart Graceful Degradation. Mikroprozessortechnik, Berlin 2 (1988) 6, S. 165
/5/ Fölsch, J.; Porep, H.-G.; Scheel, R.: Fehlertolerante Mikrorechnersysteme. Mikroprozessortechnik, Berlin 2 (1988) 6, S. 163
/6/ Kieser, H.; Bankel, M.: Einchipmikrorechner. Berlin: VEB Verlag Technik 1986
/7/ Anderer, G.: Hinweise zur Auswahl von Hochleistungs-Mikrocomputern. Elektronik, München 37 (1988) 12, S. 74
/8/ Jasukov, V.: Entwicklung sowjetischer Speicher-IS. Radio, Ferns., Elektron., Berlin 36 (1987) 11, S. 687

✉ KONTAKT ☏

VEB Forschung, Entwicklung und Rationalisierung des Schwermaschinen- und Anlagenbaus Magdeburg, Abt. Prozeßautomatisierung, Bleckenburgstraße 25, Magdeburg, 3011; Tel. 4 42 81, App. 23

TERMINE

Weiterbildungsveranstaltungen an der Ingenieurschule für Elektronik und Informationsverarbeitung „Friedrich Engels" Görlitz

- Einführung in die 16-Bit-Mikrorechentechnik
- Einsatz von Mikrorechnern zur Echtzeitverarbeitung in der Automatisierungstechnik
- Einsatz von Einchipmikrorechnern in der Automatisierungstechnik
- Informationslehrgang anwenderspezifische Schaltungen (ASIC)
- Einsatz von Arbeitsplatzcomputern in der Bürodatenverarbeitung unter dem Betriebssystem DCP
- Einsatz von CAD-Systemen in der Produktionsvorbereitung
- Konzipierung von Datenbankanwendungen und Arbeit mit unterschiedlichen Datenbankbetriebssystemen
- Softwaretechnologie
- Grundlagen der Lichtleittechnik

Beginn für alle Lehrgänge: II./III. Quartal 1990
Ort: Ingenieurschule „Friedrich Engels", Konsultations- und Weiterbildungszentrum „Mikroelektronik", PSF 648, Görlitz, 8900

Dr. Ulrich

Fachkolloquium „LAS 700-Signalmapping"

WER? Friedrich-Schiller-Universität Jena
WANN? 13. und 14. Februar 1990
WO? Jena
WAS?
- technisch-methodische Grundlagen des Signalmappings
- Hochleistungsmeßtechnik einschließlich der bis an die technisch-physikalische Grenze gehenden Erfassung von Prozeßparametern an vorgegebenen und ausgewählten Punkten
- Modellvorstellungen zur Rekonstruktion bestimmter Parameterverteilungen oder zur Generierung von Unterscheidungsmerkmalen
- komfortable, im allgemeinen bildhafte Darstellung der Mappingergebnisse und adäquate Auswerteverfahren
- Mapping für medizinische Diagnostik (EEG-EP, MEG)
- Mapping für physikalische und technologische Anwendungen (Auger-Elektronen-Spektroskopie, Inspektion technischer Oberflächen, Qualitätssicherung)

WIE? Anmeldungen für die Teilnahme senden Sie bitte bis Januar 1990 an die Friedrich-Schiller-Universität Jena, Sektion Technologie für den WGB, Technikum LAURA, Organisationskomitee „LAS-700-Signalmapping", Ernst-Thälmann-Ring 32, Jena, 6900; Tel. 8 20

Prof. Dr. sc. Entreß

Eingabemasken mit Turbo-Pascal

Bernd Matzke, Delitzsch

Die Programmiersprache Turbo-Pascal ist weit verbreitet, aber trotz aller Vorzüge sind einige Details, beispielsweise die Standardprozeduren zur Eingabe, recht spartanisch geraten. Gerade hier wären aber oft komfortablere Möglichkeiten, wie sie zum Beispiel in dBase verfügbar sind, wünschenswert.

Im folgenden sollen einige Prozeduren zur maskengesteuerten Eingabe von Werten beschrieben werden, die dem Turbo-Pascal-Programmierer ähnliche Möglichkeiten wie die dBase-Befehle GET und READ bieten.
Das Programm entstand auf einem A 7100 unter SCP 1700. Bei der Nachnutzung der Prozeduren auf der 8-Bit-Technik ergeben sich geringfügige Unterschiede, auf die im Text und im Programmlisting hingewiesen wird.
Die Übernahme des unveränderten Programms unter MS-DOS erscheint möglich, konnte aber mangels entsprechender Technik nicht erprobt werden.

Wenn eine maskenorientierte Erfassung von Daten erfolgen soll, so ist ein indizierter Zugriff auf die einzelnen Elemente der Maske. also die Vereinbarung als *Array*, erforderlich.
Da die Kursor-Steuertasten (^E, ^S, ^D, ^X) zur Auswahl eines Maskenfeldes benutzt werden sollen, müssen Tastaturabfragen zeichenweise erfolgen. Damit ist die Vereinbarung der innerhalb der Maske zu editierenden Variablen als ARRAY of CHAR oder einfacher als STRING notwendig.
Die Konvertierung aller Variablen in den gemeinsamen Datentyp STRING ist durch die entsprechende Prozedur zum Aufbau der Maske auszuführen, soll also nicht speziell für jede Maske neu programmiert werden.

Das Prinzip

Die Prozedur GET_MASKE baut ein Array auf, in dem alle zur Realisierung der Maske notwendigen Daten gespeichert werden. Vor allem sind dies die Adresse und der Typ der einzugebenden Variablen, ein Eingabefilter (ähnlich PICTURE unter dBase) sowie Daten zum Bildaufbau.
In der Prozedur wird der Inhalt der übergebenen Variable in einen String gewandelt und im Array MASKE zwischengespeichert.
Nach dem Definieren der Maske mit GET_MASKE wird mit READ_MASKE das Editieren der Bildschirmmaske eingeleitet.
Die Maske wird auf dem Bildschirm dargestellt, anschließend können Werte eingegeben werden. Das gerade zu bearbeitende Feld wird hell und invers dargestellt. Bei der Eingabe ist ein Eingabefilter wirksam.
Nach Verlassen eines Feldes wird der String in eine Variable des ursprünglichen Typs gewandelt und deren Inhalt an die entsprechende Adresse im Hauptspeicher übertragen. Der auf dem Bildschirm angezeigte String wird entsprechend der Umwandlung korrigiert.
Mit diesem Verfahren können die ansonsten durch die Typvereinbarungen auftretenden Beschränkungen umgangen und beliebige Variableninhalte zeichenweise editiert werden.
Der Editiervorgang wird zum Abschluß der Bearbeitung des letzten Feldes mit ET oder Control-X beendet.

Die Prozeduren

Dem Programmierer stehen die fünf Prozeduren CLEAR_MASKE, GET_MASKE, READ_MASKE, SAY_MASKE und ERASE_MASKE zur Verfügung. Alle anderen, hier global zugänglichen Variablen und Prozeduren sollten nicht verwendet werden. Um versehentliche Doppelvereinbarungen in anderen Programmteilen zu vermeiden und die Übersichtlichkeit zu erhöhen, erhielten die betreffenden Bezeichner den Vorsatz **m**.

Die Prozedur **CLEAR_GET** setzt den Anfangswert zum Aufbau der Maske.
Sie muß aufgerufen werden, bevor mit dem Aufbau einer neuen Maske begonnen wird.

Die Prozedur **GET_MASKE** baut das Array MASKE auf, in dem alle Daten für den Aufbau der Bildschirmmaske gespeichert werden.
Die Übergabe der Variablen erfolgt typfrei, weil durch die Prozedur Variablen unterschiedlichen Typs verarbeitet werden müssen. Da der Prozedur aber zur Umwandlung der Variablen in einen String und zum späteren Rückschreiben des eingegebenen Wertes der Typ der Variablen bekannt sein muß, wird dieser durch VARTYP mitgeteilt. Die Verwendung des richtigen Typ-Bezeichners liegt in der Verantwortung des Programmierers, eine Prüfung kann nicht erfolgen.
Den Variablen vom Typ Integer und Real sollte, aber denen vom Typ String *muß* vor Aufruf von GET_MASKE ein Wert zugewiesen werden, um Störungen des Bildaufbaus zu vermeiden.
Das Einlesen des Inhaltes erfolgt, indem auf die Adresse der typfreien Variablen V ein Varianten-Record mit drei Hilfsvariablen der Typen Integer, Real und String gelegt wird, die den typgerechten Zugriff auf die Variable ermöglichen. Folgende Werte müssen der Prozedur übergeben werden:

V	ist die zu editierende Variable.
VARTYP	kennzeichnet den Typ der Variablen V.

```
 1: PROGRAM demo; {Demonstration Eingabemasken }
 2: {$V-}                                        {Stringlaengenpruefung aus}
 3: {$C-}                                       {kein Abbruch bei Control-C}
 4: TYPE da = RECORD
 5:             bez  :STRING[30];
 6:             matnr:INTEGER;
 7:             preis:REAL;
 8:           END;
 9: VAR datensatz:da;
10:
11: {$I maske}           {File mit Prozeduren zur maskennesteuerten Eingabe}
12:
13: PROCEDURE demo_proz;
14: VAR a:STRING[20];
15:     b:STRING[5];
16:     c:INTEGER;
17:     d:REAL;
18: BEGIN
19:   a:='abcdef'; b:=''; c:=999; d:=-5.555;
20:   clear_get;
21:   get_maske(a,string_typ, 'String1:','!XXXXXXX',2, 1,SizeOf(a),20,0);
22:   get_maske(b,string_typ, 'String2:','!'       ,2,30,SizeOf(b), 5,0);
23:   get_maske(c,integer_typ,'Integer:','999'     ,4, 1,SizeOf(c), 3,0);
24:   get_maske(d,real_typ,   'Real   :','999999'  ,4,30,SizeOf(d), 6,1);
25:   { d wird auf -5.6 gerundet, da nur eine Nachkommastelle             }
26:   read_maske;                                         {Maske editieren}
27:   erase_maske;
28:   GotoXY(1,18);
29:   WriteLn('Inhalt Maske 2:');
30:   WriteLn(a); WriteLn(b); WriteLn(c); WriteLn(d);
31:   Delay(2000);
32:   say_maske;                                 { anzeigen zur Demonstration}
33: END;
34:
35: BEGIN
36:   ClrScr;
37:   WITH datensatz DO BEGIN
38:     bez:='';
39:     matnr:=12345;              {die letze Stelle wird abgeschniten da ST=4}
40:     preis:=0.0;
41:     clear_get;
42:     get_maske(bez,  string_typ, 'Bezeichnung: ',
43:                '!',1,1,SizeOf(bez),30,0);
44:     get_maske(matnr,integer_typ,'Mat.nummer : ',
45:                '9999',2,1,SizeOf(matnr),4,0);
46:     get_maske(preis,real_typ,   'Preis      : ',
47:                '9999999',3,1,SizeOf(preis),7,2);
48:   END;
49:   read_maske;
50:   GotoXY(1,8);
51:   WriteLn('Inhalt Maske 1:');
52:   WITH datensatz DO BEGIN
53:     WriteLn('Bezeichnung:',bez);
54:     WriteLn('Nummer      :',matnr);
55:     WriteLn('Preis       :',preis);
56:   END;
57:   Delay(2000);
58:   erase_maske;                         {loeschen der Maske auf dem Bildschirm}
59:   demo_proz;
60:   GotoXY(1,23);
61: END.
```

Bild 1 Demonstrationsprogramm zur Arbeit mit Eingabemasken

NAME	ist der erläuternde Name des einzugebenden Elementes, z. B. „Name“, „Bestellnummer“ o. ä.
PICT	ist ein String, der Filterzeichen zur Steuerung der Eingabe angibt.
ZEILE, SPALTE	legen Zeilen- und Spaltennummer fest, in der das Feld beginnen soll.
SIZE	gibt die mit SizeOf (var) zu ermittelnde Größe der Variablen V in Byte an.
ST, KST	Gesamtstellenzahl und Zahl der Nachkommastellen

Der Wert ST bestimmt die maximale Anzahl der einzugebenden Zeichen. Es ist dadurch möglich, die Stellenzahl des einzugebenden Wertes zu begrenzen (z. B. dreistellige Integer-Zahl). Beim Aufbau der Maske können dadurch eventuell Zeichen verloren gehen und damit ungewollt Wertverfälschungen auftreten, wenn die Stellenzahl zu klein gewählt wird (siehe Bild 1).

In der Prozedur **READ_MASKE** wird die Maske mit den Inhalten der einzelnen Elemente auf dem Bildschirm dargestellt. Anschließend werden die einzelnen Elemente editiert. Dazu stehen die folgenden Steuertasten zur Verfügung:

[Ctrl]-X oder ET	**ein Feld vorwärts**
[Ctrl]-E	**ein Feld zurück**
[Ctrl]-R	**erstes Feld**
[Ctrl]-C	**letztes Feld**
[Ctrl]-D	**Kursor ein Zeichen nach rechts**
[Ctrl]-S	**Kursor ein Zeichen nach links**
DEL und Backspace	**Zeichen links vom Kursor löschen**

Durch das Feld PICTURE im Array MASKE wird ein Eingabefilter definiert. Folgende Zeichen bewirken eine Filterfunktion:
Leerzeichen: Alle Zeichen werden unverändert übernommen.

! Umwandlung in Großbuchstaben, es werden nur Buchstaben und das Leerzeichen akzeptiert.
X Es werden nur Buchstaben oder Leerzeichen akzeptiert.
9 Es können nur Ziffern, das Leerzeichen und das negative Vorzeichen eingegeben werden.

Ist der beim Aufbau der Maske übergebene Filter-String kürzer als die maximale Stellenzahl, so wird er mit Leerzeichen aufgefüllt.
Die Prozedur wird verlassen, wenn das letzte Maskenfeld verlassen wird.

Die Prozedur **UMWAND_MASKE** wandelt den editierten String in eine Hilfsvariable des Typs der Quellvariablen um und überträgt den Inhalt der Hilfsvariablen auf deren Adresse.
Der Zugriff auf die Quellvariable wird durch einen POINTER TO INTEGER ermöglicht. Der Typ des Zeigers ist hier aber bedeutungslos, da durch die Standardprozedur MOVE keine Typprüfung von Quelle und Ziel des Datentransports erfolgt.
Er wird lediglich verwendet, da typfreie Zeiger nicht erzeugt werden können.

Die Prozedur **SAY_MASKE** stellt die Maske auf dem Bildschirm dar, ohne daß das Editie-

Bild 2 Include-Datei MASKE. PAS

```
  1: {******************************************************************}
  2: {* Include-File zur maskengesteuerten Eingabe unter Turbo-Pascal      *}
  3: {* BS: SCP1700 Rechner:A7100                           B.Matzke 1989 *}
  4: {******************************************************************}
  5: CONST m_norm = ^['[0m';              {Videosteuerzeichen normale Darstellung}
  6:       m_hinv = ^['[0m'^['[1m'^['[7m';                          {hell invers}
  7:              {bei 8-Bit Steuerzeichen durch leeren String ('') ersetzen}
  8:
  9: TYPE  m_var = (integer_typ,string_typ,real_typ);
 10:       m_str = STRING[80];        {je nach laengstem zu editierndern String}
 11:       m_typ = RECORD
 12:                 feldname,                          {Bezeichnung des Feldes}
 13:                 inhalt,                                   {Wert als String}
 14:                 picture: m_str;                            {Eingabefilter}
 15:                 typ    : m_var;                      {Typ der Variablen}
 16:                 xpos,                              {Spalte fuer Feldbeginn}
 17:                 ypos,                              {Zeile   ,,         ,,  }
 18:                 stellen,                      {Anzahl einzulesende Zeichen}
 19:                 kommast,                    {Anzahl Kommastellen bei REAL}
 20:                 groesse: INTEGER;   {Seicherbedarf der Variablen in Byte}
 21:                 posit  :^INTEGER;                  {Zeiger auf Variable}
 22:               END;
 23:       m_array = ARRAY[1..10] OF m_typ;                 {Anzahl nach Bedarf}
 24:       m_help  = RECORD CASE INTEGER OF       {dient zur beliebigen Inter-}
 25:                   1:(i: INTEGER);            {pretation einer typfreien  }
 26:                   2:(r: REAL);               {Variable                   }
 27:                   3:(s: m_str);
 28:                 END;
 29:
 30: VAR m_maske:m_array;
 31:     m_max:INTEGER;                     {Anzahl der wirklich genutzten Felder}
 32:
 33: PROCEDURE m_wandel(VAR maskenfeld :m_typ; help :m_help);
 34: {wandelt Quellvariable in String um, es wird maximal die mit STELLEN      }
 35: {festgelegte Zeichenzahl beruecksichtigt !!!                              }
 36: BEGIN
 37:   WITH maskenfeld DO BEGIN
 38:     CASE typ OF
 39:       integer_typ: Str(help.i:stellen,inhalt);
 40:       real_typ   : Str(help.r:stellen:kommast,inhalt);
 41:       string_typ : inhalt:=help.s;
 42:     END;
 43:     inhalt:=Copy(inhalt,1,stellen);
 44:     WHILE Length(inhalt) < stellen DO inhalt:=inhalt+' ';
 45:   END;
 46: END;
 47:
 48: PROCEDURE m_ls(x,y,l :INTEGER; video :m_str);
 49: {gibt an der Position x,y einen Leerstring mit l Leerzeichen aus          }
 50: {die darstellung erfolgt entsprechend dem uebergebenen video-attribut     }
 51: VAR n:INTEGER;
 52: BEGIN
 53:   GotoXY(x,y);
 54:   Write(video);
 55:   FOR n:=1 TO l DO Write(' ');
 56:   Write(m_norm);
 57: END;
 58:
 59: PROCEDURE m_writefeld(num :INTEGER; video :m_str);
 60: {Bezeichnung und Inhalt des Feldes NUM mit Video-Attribut ausschreiben    }
 61: BEGIN
 62:   IF num IN [1..m_max] THEN BEGIN
 63:     WITH m_maske[num] DO BEGIN
 64:       m_ls(xpos+Length(feldname),ypos,stellen,video);
 65:       GotoXY(xpos,ypos);
 66:       Write(feldname,video,inhalt,m_norm);
 67:       GotoXY(xpos+Length(feldname),ypos);           {Kursor an Feldanfang}
 68:     END;
 69:   END;
 70: END;
 71:
 72: PROCEDURE say_maske;                 {darstellen der Maske auf dem Bildschirm}
 73: VAR n:INTEGER;
 74: BEGIN FOR n:=1 TO m_max DO m_writefeld(n,m_norm); END;
 75:
 76: PROCEDURE erase_maske;                {loeschen der Maske auf dem Bildschirm}
 77: VAR n:INTEGER;
 78: BEGIN
 79:   FOR n:=1 TO m_max DO
 80:     WITH m_maske[n] DO
 81:       m_ls(xpos,ypos,Length(feldname)+Length(inhalt),m_norm);
 82: END;
 83:
 84: PROCEDURE clear_get;               {loeschen der gesammten Masken-Definition}
 85: BEGIN m_max:=0; END;
 86:
 87: PROCEDURE get_maske(VAR v; vartyp: m_var; name,pict :m_str;
 88:                          zeile,spalte,size,st,kst :INTEGER);
 89: VAR help_var: m_help ABSOLUTE v;        {ermoeglicht Interpretation von v}
 90: BEGIN
 91:   m_max:= m_max+1;
 92:   WHILE Length(pict) < st DO pict:=pict+' ';
 93:   WITH m_maske[m_max] DO BEGIN
 94:     feldname:=name;   picture:=pict;
 95:     stellen :=st;     kommast:=kst;
 96:     xpos    :=spalte; ypos   :=zeile;
 97:     typ     :=vartyp; groesse :=size;
 98:     posit  :=Addr(v);                          {bei 8-Bit: Ptr(Addr(v)) }
 99:     m_wandel(m_maske[m_max],help_var);                {wandeln v in String}
100:   END;
101: END;
```

```
102:
103: PROCEDURE read_maske;      {editieren der Felder INHALT des Masken-Arrays}
104: CONST ET  = ^M;                              {Tastaturcode fuer ENTER-Taste}
105:       DEL = #127;                            {     ,,          ,,   DEL - Taste}
106:       BS  = #8;                              {     ,,          ,,   Backspace  }
107: VAR num:INTEGER;
108:     kupos:INTEGER;            {Kursorposition innerhalb des Strings INHALT}
109:     t:CHAR;
110:
111:   PROCEDURE umwand_feld(num:INTEGER);
112:   {wandelt INHALT in Variable des mit TYP festgelegten Typs und          }
113:   {uebertraegt diese in die Quellvariable, INHALT wird entsprechend der }
114:   {Umwandlung korrigiert}
115:   VAR h: m_help;
116:       f: INTEGER;
117:   BEGIN
118:     IF num IN [1..m_max] THEN BEGIN
119:       WITH m_maske[num] DO BEGIN
120:         IF typ <> string_typ THEN BEGIN  {fuehrende Leerzeichen unterdr.}
121:           WHILE (Length(inhalt)>0) AND (inhalt[1]=' ') DO
122:             inhalt:=Copy(inhalt,2,Length(inhalt));
123:           IF Length(inhalt)=0 THEN inhalt:='0'; {sonst falsches Ergebnis}
124:         END;                                   {bei Val(...)            }
125:         CASE typ OF                   {Umwandeln String in Hilfsvariable}
126:           integer_typ: REPEAT
127:                          Val(inhalt,h.i,f);
128:                          IF f<>0 THEN inhalt:=Copy(inhalt,1,f-1);
129:                        UNTIL f=0;
130:           real_typ   : REPEAT
131:                          Val(inhalt,h.r,f);
132:                          IF f<>0 THEN inhalt:=Copy(inhalt,1,f-1);
133:                        UNTIL f=0;
134:           string_typ : h.s:=inhalt;
135:         END;
136:         Move(h,posit^,groesse);                {uebertragen Hilfsvariable}
137:         m_wandel(m_maske[num],h);                     {korrigiereren String}
138:       END;
139:     END;
140:   END;
141:
142:   PROCEDURE neues_feld(VAR num :INTEGER; t :CHAR);
143:   {Weiterschalten des aktivierten Feldes }
144:   BEGIN
145:     umwand_feld(num);                         {abspeichern editierter Wert}
146:     m_writefeld(num,m_norm);                 {altes Feld in normaler Darst.}
147:     CASE t OF
148:          ^E:IF num > 1 THEN num:=num-1;
149:       ^X,ET:IF num <= m_max THEN num:=num+1;
150:          ^R:num:=1;
151:          ^C:num:=m_max;
152:     END;
153:     m_writefeld(num,m_hinv);                          {neues Feld hell-invers}
154:     kupos:=1;                                          {Kursor auf Feldanfang}
155:   END;
156:
157: BEGIN {read_maske}
158:   say_maske;
159:   num:=0;
160:   neues_feld(num,^X);
161:   REPEAT
162:     Read(KBD,t);
163:     CASE t OF
164:              ^D: IF kupos < m_maske[num].stellen THEN kupos:=kupos+1;
165:              ^S: IF kupos > 1 THEN kupos:=kupos - 1;
166:  ^R,^C,^X,^E,ET: neues_feld(num,t);
167:         ' '..'z': BEGIN
168:                     IF kupos <= m_maske[num].stellen THEN BEGIN
169:                       CASE m_maske[num].picture[kupos] OF
170:                         '!':IF t IN [' ','A'..'z'] THEN BEGIN
171:                               m_maske[num].inhalt[kupos]:=Upcase(t);
172:                               Write(m_hinv,Upcase(t),m_norm);
173:                               kupos:=kupos+1;
174:                             END;
175:                         'X':IF t IN [' ','A'..'z'] THEN BEGIN
176:                               m_maske[num].inhalt[kupos]:=t;
177:                               Write(m_hinv,t,m_norm);
178:                               kupos:=kupos+1;
179:                             END;
180:                         '9':IF t IN [' ','.','-','0'..'9'] THEN BEGIN
181:                               m_maske[num].inhalt[kupos]:=t;
182:                               Write(m_hinv,t,m_norm);
183:                               kupos:=kupos+1;
184:                             END;
185:                           ':BEGIN
186:                               m_maske[num].inhalt[kupos]:=t;
187:                               Write(m_hinv,t,m_norm);
188:                               kupos:=kupos+1;
189:                             END;
190:                       END;
191:                     END;
192:                   END;
193:         DEL,BS: BEGIN
194:                     IF kupos > 1 THEN BEGIN
195:                       m_maske[num].inhalt:=
196:                         Copy(m_maske[num].inhalt,1,kupos-2)+
197:                         Copy(m_maske[num].inhalt,kupos,
198:                              m_maske[num].stellen-kupos+1)+' ';
199:                       m_writefeld(num,m_hinv);
200:                       kupos:=kupos-1;
201:                     END;
202:                   END;
203:     END;
204:     IF num IN [1..m_max] THEN
205:       GotoXY(m_maske[num].xpos + Length(m_maske[num].feldname) + kupos-1,
206:              m_maske[num].ypos);     {setzen Kursor auf korrekte Position}
207:   UNTIL num>m_max;
208: END;
```

ren eingeleitet wird. Sie wird zwar von von READ_MASKE verwendet, kann aber auch allein aufgerufen werden, wenn lediglich die Ausgabe bestimmter Informationen in definierter Form gewünscht wird.

Die Prozedur **ERASE_MASKE** löscht die auf dem Bildschirm angezeigte Maske, ohne das Array MASKE oder andere Werte zu verändern. Sie kann verwendet werden, wenn die Maske gelöscht werden soll, ohne den übrigen Bildschirmaufbau zu beeinflussen.

Übergang zu anderen Betriebssystemen

Im Programm werden **keine Tricks** verwendet, die an die Gegebenheiten eines speziellen Rechners oder Betriebssystems gebunden sind.
Probleme können sich lediglich durch die Verwendung der Bildschirmsteuerzeichen ergeben, die hier benutzt werden, um den gerade zu editierenden Wert hervorzuheben. Gerade in dieser Frage bestehen prinzipielle Unterschiede zwischen den Geräten der 8- und 16-Bit-Technik.
Es ist empfehlenswert, bei Verwendung anderer Betriebssysteme bzw. Rechner auf die Nutzung der Bildschirmattribute zu verzichten. Die Markierung des gerade zu editierenden Elementes erfolgt ohnehin durch den Kursor. Die beiden Steuerzeichenfolgen in der Konstantenvereinbarung werden dazu einfach durch zwei Leerstrings ersetzt, dies erspart das Streichen der entsprechenden Bezeichner im gesammten Quelltext.
Die Portabilität des Programms wird durch den Wegfall der Bildschirmsteuerzeichen erhöht.
Die einzige echte Quelltextänderung wird bei Nutzung der 8-Bit-Technik erforderlich. Die Zuweisung der Variablenadresse zum Zeiger POSIT erfolgt hier durch PTR(ADDR(v)) anstelle von ADDR(v). Zur Nutzung der Prozeduren unter Turbo-Pascal 4.0 bzw. 5.0 kann ich leider noch keine Aussagen treffen.

TERMINE

Der Bezirksfachausschuß Mikroelektronik beim Bezirksvorstand Berlin der KDT führt im ersten Halbjahr 1990 Spezialkonsultationen zu folgenden Themen durch:

14. Febr. Rechnergestütztes Informationssystem Steuerungstechnik – Ein Werkzeug für den Hard- und Softwareentwickler
21. März Stand und Trend der digitalen Tonsignaltechnik
11. April Probleme der computergesteuerten Bilderkennung und -vorbereitung an einem Beispiel aus der Landwirtschaft
9. Mai Probleme der Datensicherung auf PC
20. Juni Anwendungsumgebungen von Leiterplatten-CAD-Lösungen

Änderungen sind dem Bezirksfachausschuß Mikroelektronik vorbehalten. Die Teilnahme an den Konsultationen bedarf keiner Anmeldung und ist kostenlos. *Die Konsultationen finden jeweils 14.00 Uhr im Haus der KDT, 1086 Berlin, Kronenstraße 18, statt.* *Driese*

ROM-Erweiterungen für IBM-PCs

Jens Immig
Akademie der Wissenschaften der DDR, Institut für Kosmosforschung

Einer der wesentlichen Vorteile der IBM PC/ XT,/AT und ihrer Kompatiblen ist der modulare Aufbau ihres Systems, der ein problemloses Wechseln oder Ausbauen der Rechnerkonfiguration zuläßt. So ist es auch möglich, daß Rechner dieser Familie mit Modulen ausgerüstet werden können, die erst zu späteren Zeitpunkten entwickelt wurden.

Funktion des BIOS

Zur Schaffung einheitlicher Schnittstellen zu der vom Hersteller verwendeten Hardware wird der Hardware-Kern des Rechners (der Prozessor) in einen Software-Kern eingebettet. Dieser Software-Kern ist das *Basic Input Output System* (BIOS). Das BIOS befindet sich zum größten Teil fest auf ROM-Bausteinen, welche sich auf der Grundplatine des Rechners befinden.
Das Firmware-BIOS (BIOS der Grundkonfiguration des Rechners) steht im ROM ab Adresse FE000H, der Prozessor startet ab Adresse FFFF0H. Dort findet er die Information, an welcher Stelle er fortfahren muß /1/.
Meist ist es möglich, auf der Grundplatine weitere BIOS-Routinen auf ROM-Steckplätzen unterzubringen, wobei die Steckplätze auf der Grundplatine festen Adressen zugeordnet sind.
Die BIOS-Eintrittsstelle ist eine im RAM ab Adresse 00000 organisierte Liste, in welcher die Anfangsadressen der aktuellen Softwareinterruptroutinen, der sogenannten BIOS-Interrupts steht. Im Verlauf der Grundinitialisierung wird hier eine minimalen Ansprüchen genügende Liste der Serviceroutinen aufgebaut, die im ROM-BIOS enthalten sind. Um hier mehr Komfort zu gewährleisten, wird meist außer dem DISK-Manager (DOS), welcher einige der anfallenden Routinearbeiten zum Zugriff auf einen Datenträger (z. B. Disk) übernimmt, auch ein BIOS.COM geladen (RAM-resident).
Diese Systemroutinen sind die Standardgerätetreiber. Die Gerätetreiber stellen die Schnittstelle zwischen dem Betriebssystem und den Geräten dar.

Wozu ROM-Erweiterungen?

Nun kann das Original-ROM-BIOS eines Rechners unmöglich die Unterstützungsprogramme für zukünftige Geräteentwicklungen bereitstellen. Um dennoch solche Geräte an den Computern stationieren zu können, wurde von den PC-Geräteentwicklern eine kluge Entwurfsstrategie für die PCs geschaffen. Es gibt erstens die bereits oben genannte Möglichkeit, auf ROM-Steckplätzen der Grundplatine weitere BIOS-Routinen unterzubringen. Zweitens wird beim Laden des Betriebssystems (MS-DOS/PC-DOS, im folgenden kurz mit DOS bezeichnet) im Rootverzeichnis (Grundverzeichnis) des Ladegerätes eine Konfigurationsdatei mit dem Namen CONFIG.SYS gesucht, die Angaben zum Einstellen von Systemparametern und zu speziellen Geräten enthält /2/. Innerhalb dieser Datei können durch die Anweisung DEVICE (siehe /2/, S. 41) neue Gerätetreiber in die Behandlung durch DOS einbezogen werden. Hier können neben den zum DOS gehörenden Treibern sowohl die von Herstellern von Spezialgeräten mitgelieferten Treiber als auch eigene Treiber eingebunden werden. Diese Treiber müssen die vorhandenen BIOS-Routinen nutzen oder direkt die entsprechenden Bausteine über Ports und Adressen programmieren /2/. (Der Treiber CDRIVE stellt z. B. ein Diskettenlaufwerk auf CP/M-Format ein.) Auch während der Arbeit am Rechner können neue Gerätetreiber geladen werden.
Drittens sind beim PC/XT,/AT und bei kompatiblen Rechnern genau festgelegte Speicherbereiche innerhalb des direkt adressierbaren Speicherraums für die Schaffung von Erweiterungen reserviert. Diese Adressbereiche (C0000H bis EC000H) sind auch durch ROM-Erweiterungen nutzbar. ROM-Erweiterungen sind auf ROM-(oder EPROM-)Bausteinen untergebracht, die auf den steckbaren Adaptern sitzen. Werden diese Adapter in die dafür vorgesehenen Steckverbinder auf der Grundplatine gesteckt, so erfolgt eine direkte Verbindung mit dem Rechnerbus.
ROM-Erweiterungen sind in der Regel Unterstützungsprogramme für neue Peripherie. Als Beispiele seien hier das Hilfsprogramm für ein eingebautes Festplattenlaufwerk oder das Video-ROM von hochauflösenden (High Resolution) Color-Grafikadaptern genannt.

Das Prinzip von ROM-Erweiterungen

Das Prinzip der BIOS-Erweiterung durch zusätzliche Komponenten beruht darauf, daß diese Komponenten BIOS-Interrupts emulieren, das heißt durch Eintragen der eigenen Adresse (4 Byte) in die Interrupttabelle eine BIOS-Funktion vollständig übernehmen. Folgende Aufgaben können durch ROM-Erweiterungen übernommen werden:
- Konfigurationstest und Initialisierung der dazugehörigen Hardware
- Umlenken von BIOS-Interrupts
- Meldung für Benutzer auf dem Bildschirm.

ROM-Erweiterungen werden beim Startprozeß des Rechners mit berücksichtigt. Um sie zu finden, sucht der Standard-ROM im Adreßbereich von der absoluten Adresse C8000H bis E0000H alle 2 KByte nach dem Kennzeichen für ROM-Erweiterungen (55AAH). Wird ein solches Kennzeichen gefunden, so übergibt das Startprogramm die Kontrolle an die ROM-Erweiterung, damit diese alle zu ihrer Integration nötigen Maßnahmen veranlassen kann. Ist die Initialisierung abgeschlossen, gibt die Erweiterung die Kontrolle an das Startprogramm zurück, das in seiner Aufgabe an dieser Stelle fortfährt /3/.
Die ROM-Erweiterungen haben gegenüber den ladbaren Treibern den Vorteil, daß sie keinen zusätzlichen RAM benötigen (es sei denn, das Programm der Erweiterung veranlaßt ein Beschreiben des RAMs). Sie werden immer dann verwendet, wenn ein Peripheriegerät schon während des Startprozesses des Rechners initialisiert werden soll (z. B. Grafikadapter).

Aufbau von ROM-Erweiterungen

Soll eine ROM-Erweiterung zum Beispiel als Unterstützungsprogramm oder als Initialisierung für ein selbstentwickeltes Peripheriegerät geschaffen werden, so müssen einige wichtige Details beachtet werden.

Je[illegible] in de[illegible] Ingenie[illegible] k-trotechnik [illegible] ie Berlin [illegible] Spezialisierungsrichtung [illegible] o-rechentechnik. Seit 1988 arbeitet er an [illegible] ut für Kosmosforschung der Akademie [illegible] n-schaften als Forschungsingenieur. Hier befaßt er sich mit Hard- und Software für 16-Bit-Mikrore[illegible] n-nersysteme.

Der Bereich von der absoluten Adresse C0000H bis C8000H muß für die Video-ROMs frei gehalten werden. Dieser Bereich wird nur auf Treiberprogramme für Grafikkarten wie EGA (Enhanced Graphics Adapter), VGA (Video Graphics Array) o. ä. untersucht /4/. Ist eine Festplatte angeschlossen (heute Standardausrüstung für IBM PCs und Kompatible), so befindet sich die dafür notwendige ROM-Erweiterung des Festplattencontrollers auf der Adresse C8000H. Wurde ein zweiter Festplattencontroller installiert, so ist seine Treiberroutine bei CA000H zu finden. Die Hilfsprogramme für Festplatten umfassen jeweils 8 KByte, so daß man eine eigene ROM-Erweiterung erst auf die Adresse CC000H oder am günstigsten erst ab D0000H bzw. dahinter legen sollte.
Da der Bereich von der Adresse D0000H bis EC000H für Erweiterungen (z. B. auch RAM-Erweiterungen) freigehalten wurde, ist es nie ganz ausgeschlossen, daß man mit Routinen oder Bereichen anderer Module in Konflikt gerät. Deshalb sollte man sich zunächst genau mit der Konfiguration seines Rechners vertraut machen.
Damit eine Adapter-ROM-Erweiterung von der Startroutine beachtet wird, muß sie wie folgt aufgebaut sein:
Sie muß bei einer physischen Adresse beginnen, die sich aus ADR = C8000H + (N * 2 KByte) (N = 1, 2, 3...) ergibt (maximal E0000H). Am Beginn muß die Kennung für ROM-Erweiterungen stehen. Das sind die beiden aufeinanderfolgenden Bytes 55H AAH.
Das dritte Byte ist ein Längenanzeiger. Der hier stehende Wert gibt Auskunft über die Anzahl der 512-Byte-Blöcke, die Bestandteil der aktuellen ROM-Erweiterung sind. Dabei ist wichtig, daß auch ein angefangener 512-Byte-Block am Ende des ROM-Programms (vollständig!) als Bestandteil der Erweiterung anzusehen ist. Von der Startroutine des Rechners wird diese Information für einen Prüfsummentest benötigt, bei dem festgestellt wird, ob es sich um eine korrekte, vollständige ROM-Erweiterung handelt. Bei diesem Test werden alle Bytes des so definierten ROM-Moduls (einschließlich der Kennung 55AAH) modulo 100H addiert. Ist die ermittelte Summe gleich Null, so wird durch CALL FAR auf das Byte mit dem Offset 3 des ROMs gesprungen, bei dem das lauffähige Maschinenprogramm beginnen muß. Ist die Summe ungleich Null, so wird eine Fehlermeldung ausgegeben. Es ist also wichtig, innerhalb der ROM-Erweiterung ein Ausgleichsbyte zu setzen, welches die Summe (modulo 100H) über alle Bytes der definierten 512-Byte-Blöcke gleich Null werden läßt.
Bei den ROM-BIOS-Erweiterungen auf den festen Steckplätzen ist die Länge jeweils 8 KByte (2664 oder 2764), das dritte Byte wird hier nicht benutzt. Die ROM-Erweiterung muß durch ein RETURN FAR verlassen werden (in Maschinencode CBH), damit die Startroutine des Computers nach dem Abarbeiten der Er-

weiterung an der Stelle fortfahren kann, von der sie kam.

Das Initialisierungsprogramm kann alle Prozessorbefehle der 8086-Familie beinhalten. Weiterhin können alle Interrupttypen verwendet werden, die direkt die BIOS-Routinen des Original-ROM-BIOS ansprechen. Beachtet werden muß, daß die DOS-Systemschnittstelle (alle Interrupts ab 20H) von einer ROM-Erweiterung nicht verwendet werden kann, da beim Abarbeiten des ROMs nach dem Rechnerstart das Betriebssystem (MS-DOS/PC-DOS) noch nicht geladen ist und somit die DOS-Funktionen noch nicht zur Verfügung stehen.

Programmerzeugung

In Bild 1 ist eine kurze Assemblerroutine zu sehen, welche nach dem Übersetzen mittels EPROM auf die Adresse D0000H gelegt werden soll. Durch diese Routine wird nach dem Speichertest des Original-BIOS eine kurze Ausschrift auf dem Bildschirm sichtbar.

Die ROM-Erweiterung muß immer so aufgebaut sein, daß beim vierten Byte das lauffähige Programm beginnt, da von der ROM-Suchroutine stets an diese Stelle gesprungen wird.

Testen einer ROM-Erweiterung

Es ist empfehlenswert, in der Versuchsphase statt der EPROMs pinkompatible statische RAM-Bausteine zu stecken, die beispielsweise durch einen Debugger oder noch einfacher durch ein kleines Turbo-Pascal-Programm geladen werden können. Bei einem Warmstart des Rechners wird die auf ihnen abgelegte Information nicht gelöscht, da sie keine Refreshzyklen benötigen und in den Erweiterungsbereichen durch das Original-ROM-BIOS kein Speichertest vorgenommen wird. Man kann sich dadurch beim Eintesten der ROM-Erweiterung und der dazugehörigen Peripherie die viele Arbeit des Beschreibens und wieder Löschens von EPROMs ersparen und auf diese erst zurückgreifen, wenn man die endgültige Programmvariante gefunden hat.

```
prog    segment
        assume  cs:prog,ds:prog

rom:    db      55h,0aah,01h                 ;Kennung fuer ROM-
                                             ;Erweiterung

start:  jmp     start1                       ;Beginn bei Start1
text:   db      'Test fuer ROM-Erweiterung !'
        db      24h

start1: mov     dx,0d000h                    ;Text in Segment mit
        mov     es,dx                        ;Adresse D0000h
        mov     dx,offset text               ;Datenoffset auf Textanfang
        mov     si,dx

        mov     ah,2                         ;Setzen der Curser-
        mov     dh,3                         ;position
        mov     dl,2                         ;Zeile=3
        mov     bh,0                         ;Spalte=2
                                             ;Seite=0

marke:  mov     ah,2                         ;Aktuelle Curserposition
        int     10h                          ;einstellen

        mov     al,es:[si]                   ;AL=aktuelles Textzeichen
        cmp     al,24h                       ;Textende erreicht ?
        je      ende                         ;wenn 'Ja', dann Programmende
        mov     bl,0fh                       ;nein, dann Ausgabe des
        mov     bh,0                         ;Zeichens an aktueller
        mov     cx,1                         ;Curserposition durch
        mov     ah,9h                        ;BIOS-Interrupt
        int     10h
        inc     si                           ;Datenoffset um 1 erhoehen
        inc     dl                           ;Curserposition 1 nach rechts
        jmp     marke                        ;zurueck zu MARKE

ende:   mov     cx,0ah                       ;kurze Warteschleife
m1:     mov     ax,0ffffh
m2:     dec     ax
        cmp     ax,0
        jnz     m2
        loop    m1

        retf                                 ;zurueck zur Startroutine

        db      98h                          ;Ausgleichsbyte fuer
                                             ;Summentest
        db      418 dup(0ffh)                ;Auffuellen des 512-byte-
                                             ;Blocks
prog    ends

        end     start
```

◀ *Bild 1 Assemblerprogramm*

Literatur

1/ Brors, I.: Maschinensprache des IBM-PC in der Praxis. Heidelberg: Dr. Alfred Hüthig Verlag 1986

2/ Hübener, J.: MS-DOS. Berlin: Verlag Technik 1988

3/ Norton, P.: Programmierhandbuch für den IBM PC. Braunschweig: Friedr. Vieweg & Sohn Verlagsgesellschaft 1987, S. 62

4/ IBM PC/AT Technical Reference. IBM Corporation 1986

5/ Jobst; Lutz; Selder: Intel 16/32 Bit Assemblerhandbuch. Kissing: Interest-Verlag 1987

6/ Der IBM PC und seine Kompatiblen. Mikroprozessortechnik, Berlin 2 (1988) 8, S. 234

7/ Smode, D.: Das große MS-DOS-Profi-Arbeitsbuch. München: Franzis-Verlag 1987

✉ KONTAKT ☏

Akademie der Wissenschaften der DDR, Institut für Kosmosforschung, Abt. Nachrichtensysteme 2, Rudower Chaussee 5, Berlin, 1199; Tel. 6 74 50 75

Im Test: C-Compiler für CP/M

Uwe Schulze, Berlin

Wollen Sie sich für einen C-Compiler für CP/M entscheiden – etwa um eine Sprache zu lernen, die sich in der professionellen Programmierung breit durchgesetzt hat (neuere Versionen von Wordstar und dBase sowie OS/2 sind größtenteils in C geschrieben) –, so werden Sie feststellen, daß es hier keinen Quasi-Standard-Compiler gibt, sondern eine ganze Reihe gleichwertiger Pakete. Welches davon Verwendung finden soll, hängt in starkem Maße von den Anforderungen der Anwendung ab: Liegt das Primat auf kompaktem Code oder schnellen Ausführungsgeschwindigkeiten, ist eine Realarithmetik erforderlich, wird Assembler-Zwischencode benötigt etc.? Gerade weil unter CP/M alles handmade ist, ist C für den Ein- oder Umsteiger gut faßbar und verständlich.

Der Test

Zur Bewertung von Compilern sind ihre Ausstattung oder die vollständige Aufnahme des Sprachstandards, der Entwicklungskomfort und das Laufzeitverhalten von Interesse. Der Programmierer sieht sich mit einer Vielzahl von Compiler-, Assembler- und Linkerläufen konfrontiert, so daß die Turn-around-Zeiten (vollständiger Testumlauf) am besten Auskunft über die Bedingungen der Programmentwicklung geben. Um die Abhängigkeit von der Anzahl der Pässe zu zeigen (die jeweiligen Übersetzungskomponenten müssen geladen und die Zwischencodes gespeichert werden), wurde das „leere" Programm *main* mit einer SUBMIT-Datei in eine COM-Datei übersetzt:

```
main ()
    {
    }
```

Von Interesse ist auch die Codegröße dieses kleinsten Programms (Tafel 1), zeigt sie doch den beim Linken entstehenden Overhead (was beispielsweise für ROM-fähigen Code von Bedeutung ist). Einfluß darauf hat aber auch der Programmierer, wenn er die Bibliotheksfunktionen als Include, durch Cuttern oder durch Linken einfügt. Für die Testbeispiele wurden die Bibliotheken jeweils zugelinkt, was die variabelste Methode ist, nicht aber notwendigerweise die kleinsten Programme erzeugt.

Als Standardbenchmark gilt das Primzahlensieb des Eratosthenes, das die Anzahl der Primzahlen zwischen 3 und 16381 durch Streichen der Vielfachen in einem Feld ermittelt. Die geraden Zahlen werden dabei ausgeschlossen, so daß ein Feld der Dimension 8190 benötigt wird. Das Programm enthält also eine Vielzahl von Speicherzugriffen. Das in Bild 1 dargestellte Listing trägt der Tatsache Rechnung, daß die Speicherklassen (Static, Register) nicht durchgehend enthalten sind. Somit stellt es einen auf allen getesteten Compilern lauffähigen Standard dar.

Als zweiter Test dient die rekursive Protokollierung der Fibonacci-Funktion im Integer-Bereich (Bild 3). Damit läßt sich das Zeitverhalten bei Unterprogrammaufrufen und Parameterübergaben bewerten. Der Datentyp *Long* ist wiederum nicht durchgehend enthalten.

Zum Schluß soll noch ein Blick auf die Bild-

```
#define TRUE 1
#define FALSE 0
#define SIZE 8190
#define SIZEPL 8191

char flags[SIZEPL];
int i,prime,k,count,iter;

main()
{
  printf("Hit return to do 10 iterations: ");
  getchar();

  for (iter = 1; iter <= 10; iter++) {
      count = 0;
      for (i = 0; i <= SIZE: i++)
          flags[i] = TRUE;
      for (i = 0; i <= SIZE; i++) {
          if (flags[i]) {
             prime = i + i + 3;
             k = i + prime;
             while (k <= SIZE) {
                   flags[k] = FALSE;
                   k += prime;
                   }
             count++;
          }
      }
  }
  printf("\7\n%d primes.\n",count);
}
```

Bild 1 Primzahlensieb des Eratosthenes in C

Tafel 1 C-Compiler im Vergleich

Compiler	Assembler-zwischencode	Ar[illegible]	Real-Arithmetik	Übersetzungszeit MAIN in min	Codegröße MAIN
BDS-C				[illegible]	(16 records)
Supersoft-C	ja	4	nein	1:32	4 KByte (23 records)
Eco-C	ja	3	ja	3:02	12 KByte (82 records)
Aztec-C	ja	3	ja	1:23	4 KByte (26 records)
Wh[illegible]	ja	5	ja	3:58	8 KByte (55 records)

Tafel 2 Laufzeitverhalten

Compiler	Primzahlensieb Laufzeit in min	[illegible]zeit in min	[illegible]	[illegible]funktion in min	[illegible]
B[illegible]	25	0:41	4 KByte	0:18	1:13
Supersoft-C	0:34	1:58	12 KByte	0:12	1:13
E[illegible]	0:33	4:17	28 KByte	0:17	1:40
A[illegible]	0:37	2:04	10 KByte	0:22	1:51
W[illegible]-C	0:43	6:50	16 KByte	0:13	1:58
[illegible]	0:39	0:10 (COM) 0:01 (Mem)	10 KByte	0:45	0:35

schirmausgabe geworfen werden. Dazu werden 32000 Sterne mit PRINTF auf den Bildschirm geschrieben. Die relativ langsame Ausgabe liegt in der Tatsache begründet, daß die C-Compiler die BDOS-Schnittstelle benutzen, während Turbo-Pascal bekanntlich direkt auf das BIOS zugreift (Tafel 2).
Alle Tests wurden auf einer 2,5-MHz-CPU unter CP/A vorgenommen, woraus sich die Differenzen zu /2/ erklären. Auch unterscheiden sich die Compilerversionen. Zum besseren Vergleich gibt es die gleichen Tests auch in Turbo-Pascal (Bild 2 und Tafel 2).

Die Kandidaten

BDS-C, BD-Software, Version 1.5

Der BDS-C-Compiler ist leistungsfähig und einfach handhabbar. In nur zwei Schritten erzeugt er ein COM-File. Die Übersetzung des Primzahlensiebes sieht wie folgt aus:

CC Sieve
CLINK Sieve

Alternativ kann auch der Linker L2 verwendet werden. Das Einbinden benötigter Bibliotheken erfolgt automatisch. Als einziger der untersuchten Compiler erzeugt er keinen Assembler-Zwischencode. Daraus resultieren schnelle Übersetzungszeiten, aber die Möglichkeit, Module aus anderen Sprachen mit dem Linker einzubinden, entfällt, da der Zwischencode nicht Microsoft-compatibel ist. Der Compiler selbst ist in 8080-Maschinencode geschrieben – nicht in C. Die Grundausstattung des ursprünglich für die Systementwicklung gedachten Compilers ist eher spartanisch: Die Speicherklassen **STATIC** und **REGISTER** fehlen ebenso wie die Datentypen **FLOAT** und **LONG**. Aber es existieren Zusatzpakete (sämtlich im C-Quelltext), die keine Wünsche offen lassen: ein zweiter verbesserter Linker, ein Overlaymanager, ein ASM-Präprozessor, ein Debugger sowie ein Integer- und Floating-Point-Package, das die fehlenden Datentypen als Funktionen zur Verfügung stellt.
Zur Optimierung des Primzahlensiebes werden die Daten auf eine feste Adresse statt auf den Stack gelegt (Option: –e1000 für EXTERNAL) und damit eine erstaunliche Geschwindigkeitsverbesserung erzielt (25 statt 53 s).

Supersoft C, Supersoft Inc., Version 1.1.25

Der Supersoft-C-Compiler erzeugt in zwei Pässen Assemblercode. Die Bibliotheken können als C-Quelltext eingefügt oder als Objektcode zugelinkt werden; in jedem Falle sind sie explizit anzugeben. Mit der Compilerdirektive #ASM ... #ENDASM kann Assemblercode direkt in den Quellcode einge-

```
program sieve;
const
   size = 8190;
var
   flags: array[0..size] of boolean;
   i,prime,k,count,iter: integer;
   c:char;
begin
   writeln('Hit return to do 10 iterations');
   read(c);
   for iter:=1 to 10 do begin
   count:=0;
   for i:=0 to size do
        flags[i]:=true;
   for i:=0 to size do
        if flags[i] then begin
           prime:=i+i+3;
           k:=i+prime;
           while k <= size do begin
                 flags[k]:=false;
                 k:=k+prime;
           end;
           count:=count+1;
        end;
     end;
   writeln(count,' primes');
end.
```

Bild 2 Primzahlensieb des Eratosthenes in Pascal

```
int i;
#define NUMBER 22
main()
{
unsigned value,fib();
printf("\n Fibonacci series.\n");
getchar();
for (i=1; i<=NUMBER; i++)
{ value=fib(i);
  printf("\nfibonacci(%d)     =%u.\n",i,value);
}
}
unsigned fib(j)
int j;
{
if (j>2) return (fib(j-1) + fib(j-2));
 else return(1);
}
```

```
{@A-}
program fibo;
var
i:integer; c: char;

function fibo(n: integer):integer;
begin
   if n <= 2 then fibo := 1
   else fibo:=fibo(n-1)+fibo(n-2);
end;

begin
   writeln('Fibonacci series');
   read(c);
   for i:=1 to 22 do
   writeln(fibo(I):10);
end.
```

Bild 3 Protokollierung der Fibonacci-Funktion in C und Pascal

fügt werden. Ein typischer Übersetzungslauf sei am Beispiel des Primzahlensiebes gezeigt:

CC SIEVE.C
C2 SIEVE.COD
M80 = SIEVE.ASM
L80 SIEVE,C2RT,FORMATIO,STDIO, ALLOC,CRUNT2,SIEVE/E/N

In Abweichung zum C-Standard ist die Speicherklasse **STATIC** nicht enthalten, lange Integer- sowie Gleitkomma-Arithmetik fehlen.
Die Verwendung von Registervariablen zur Verbesserung des Primzahlensiebes erzeugt zwar veränderten Code, führt aber zu keiner Laufzeitverbesserung.

EcoC, Ecosoft Inc., Version 3.00

In drei Pässen erzeugt der Eco-Compiler Assemblercode. Dabei kommt der Code-Generation-Paß nicht mit 48 KByte TPA aus! Die Handhabung ist einfach; ein Übersetzungslauf hat folgende Syntax:

CP SIEVE
M80 = SIEVE
L80 SIEVE,SLIBCO,SIEVE/E/N

Die Compiler-Pässe liegen als getrennte COM-Files vor und können auch einzeln aufgerufen werden. **STATIC** und **REGISTER** sind als Speicherklassen nicht enthalten, aber es existieren die Datentypen **FLOAT** und **LONG**.

Aztec-C, Manx Software Systems, Version 1.06d

Dieser Compiler ist sehr komfortabel ausgestattet. Er stellt Assemblercode zur Verfügung, der mit einem Standardassembler oder dem mitgelieferten AS.COM weiterverarbeitet werden kann. Z80-spezifischer Maschinencode wird mit DB-Anweisungen eingefügt. Zur Übersetzung dient folgende Kommandofolge:

CZ SIEVE
AS SIEVE
LN SIEVE.O C.LIB

Der Compiler orientiert sich an Unix-C. Bis auf Bitfelder wird der gesamte Sprachumfang von /1/ angeboten. Neben Bibliotheken für drei verschiedene Speichermodelle wird noch eine Version zur Erzeugung ROM-fähigen Codes geliefert. Stärke des Compilers ist die Real-Arithmetik. Die Datentypen **LONG**, **FLOAT** und **DOUBLE** stehen zur Verfügung. Auch die Speicherklassen **STATIC** und **REGISTER** sind implementiert. Die Bibliotheksfunktionen liegen sämtlich als C- oder Assemblerquelle vor. Ferner existiert noch eine Reihe nützlicher Funktionen im Quellcode (zum Beispiel Winkelfunktionen, Quicksort, Zufallszahlengenerator).
Gegenüber der Vorgängerversion wurde vor allem die Handhabung verbessert und die Codegröße verringert.

Native-C, Whitesmiths Ltd.

Vor dem ersten Übersetzungslauf hat der Programmierer seine Befähigung nachzuweisen, indem er die sämtlich als Objektcode vorliegenden Compilerkomponenten mit LINK verbindet und lauffähig macht. Damit sind die Schwierigkeiten noch nicht beseitigt.

Der Parser-Paß belegt als COM-File 50 KByte (!) und kann nur mit einem System-Subset betrieben werden. Ein ohne Stapeldatei kaum zu bewältigender Übersetzungslauf hat folgendes Aussehen:

PP -X -I A: -O SIEVE.TM1 UTEXT.H SIEVE.C
P1 -BO -N8 -O SIEVE.TM2 SIEVE.TM1
P280 -O SIEVE.S SIEVE.TM2
AS80 SIEVE.S
LINK -EB__MEMORY -ED__EDATA-TBOX100 -HTR -O SIEVE.COM CRTS.80 SIEVE.O LIBC.80

Dabei fällt der Linker durch seine Langsamkeit auf. In 4:43 min, die er für das Binden des Primzahlensiebes benötigt, hat der BDS-Compiler das Programm 6½mal vollständig übersetzt! Der Sprachumfang ist vollständig aufgenommen (76 KByte Bibliotheksgröße) und der Compiler eng an UNIX-Versionen angelehnt. **DOUBLE**, **FLOAT** und **LONG** sind als Datentypen ebenso vorhanden wie die Speicherklassen **REGISTER**, **STATIC** und **EXTERNAL**. Unverständlich aber die unüblichen Namen für Standardfunktionen: PRINTF heißt hier PUTFMT, und auch die Formatparameter halten sich nicht an die üblichen Konventionen.

Das Fazit

Am besten schneidet der BDS-C-Compiler ab. Sowohl durch Verarbeitungs- und Übersetzungsgeschwindigkeit als auch durch einfache Handhabung empfiehlt er sich. Mit den Zusatz-Tools wird er allen Anforderungen gerecht.
Positiv beurteilt werden kann, was Handhabung und Ausrüstung betrifft, auch der Supersoft-C-Compiler – solange keine Real-Arithmetik benötigt wird. Die schnellste Real-Arbeit leistet Aztec-C, auch Ausstattung und Handhabung lassen ihn empfehlen.
Whithesmiths-C realisiert den vollständigsten Sprachumfang, aber der allzu großzügige Umgang mit Speicherplatz und Übersetzungszeit ist kaum zu akzeptieren.
Nicht zufrieden stellen können auch die teilweise wesentlich zu großen COM-Files bei einigen Compilern. Hier ist der Weg des Linkens mit den Bibliotheksfunktionen nicht in jedem Falle zu empfehlen. Eventuell sollten die benötigten Funktionen im Quellcode eingefügt werden.

Literatur

/1/ Kernighan, B. W.; Ritchie, D. M.: The C Programming Language. Prentice Hall 1978
/2/ Gilbreath, J.; Gilbreath, G.: Eratosthenes Revisited. BYTE Publications Inc., January 1983
/3/ Stuebs, M.: Computersprache C: Der flinke UNIX-Sprößling. CHIP, München 5 (1983) 5, S. 154
/4/ Horn, T.: Programmierung in C. Mikroprozessortechnik, Berlin 1 (1987) 1–6

Farbgrafikadapter für K 1520-Systeme – klein und vielseitig

Kathrin Lajow, Wolfram Kammer, Wolfgang Spindler
VEB Elektronische Bauelemente Teltow

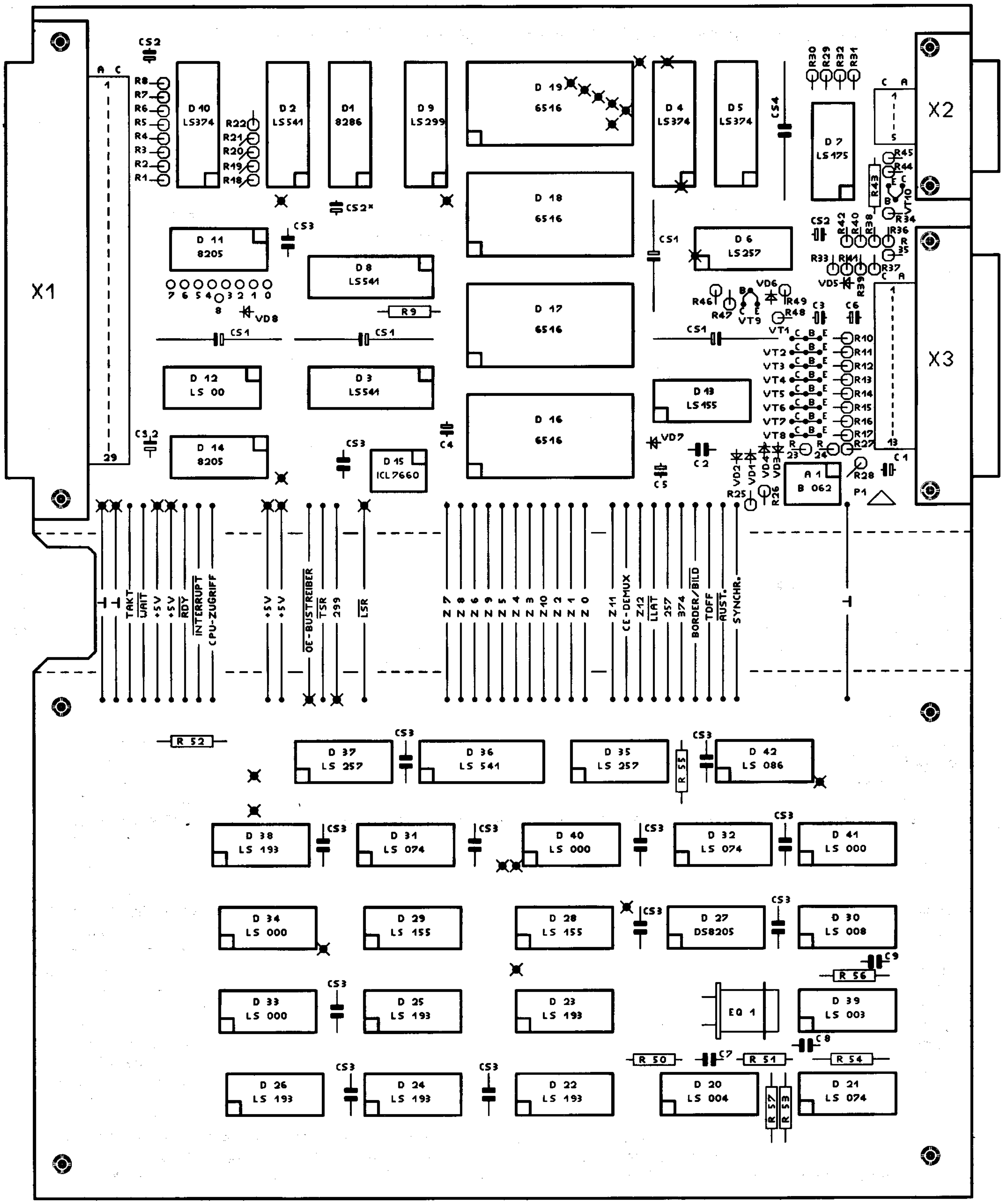
Grafische Darstellungen in Farbe auf dem Bildschirm gehören wohl zu den nobelsten Arten der Kommunikation zwischen Computer und Bediener. Wenn es nicht gerade ein hochauflösendes Gerät für die CAD-Welt sein soll, dann bleibt der Aufwand für die dazu notwendige Hardware recht gering, wie dieser Artikel zeigt.

Bevor es richtig losgeht, muß hier gesagt werden, daß dieser Farbgrafikadapter nicht ohne weiteres an jeder x-beliebigen K 1520-Konfiguration betreibbar ist. Der Adapter liefert nämlich einen Systemtakt von 3,5 MHz auf den Systembus, und alle übrigen Baugruppen des Rechners müssen mit eben diesem Takt arbeiten – sonst geht nichts. Wenn diese Bedingung (es ist glücklicherweise die einzige, die etwas aus dem gewohnten Rahmen fällt) erfüllt ist, dann steht den farbenprächtigen Bildern auf dem Monitor nichts mehr im Wege. Der Adapter erzeugt ein normgerechtes Fernsehbild mit 15,625 kHz Zeilenfrequenz und 50 Hz Bildfrequenz ohne Zeilensprung. Er liefert an seinem Monitorausgang die vier digitalen Signale *ROT*, *GRÜN*, *BLAU* und *EXTRAHELL* sowie ein Synchronsignal (alles mit TTL-Pegel) und zusätzlich ein analoges BAS-Signal, bei dem die Farben in Grauwerte gewandelt sind. Notfalls kann man also auch einen Schwarzweiß-Fernseher anschließen.

Die technischen Daten

Das Bildfeld, welches der Adapter auf den Bildschirm schreibt, ist in horizontaler Richtung 256 Pixel und in vertikaler Richtung 192 Pixel groß. Jedes Pixel ist separat schreib- und lesbar. Das Bildfeld füllt nicht den gesamten Bildschirm aus. Bis zu den Bildschirmrändern ist es mit einem etwa 20 Pixel breiten Rand umgeben, da erfahrungsgemäß die Bildschärfe in den Ecken zu wünschen übrig läßt. Die Pixel des Bildfeldes schalten zwischen dem sogenannten Vordergrund (Pixel gesetzt) und dem Hintergrund (Pixel gelöscht) um. Die Farben von Vordergrund, Hintergrund und Rand sind unabhängig voneinander einstellbar. Einen Alphanumerikmodus mit Zeichengenerator usw. gibt es bei diesem Adapter nicht. Wer auf diesem Bildschirm Text anzeigen will, muß deshalb im Gegensatz zu alphanumerischen Bildschirmen jedes Zeichen aus einzelnen Pixeln zusammensetzen. Das ist aber gar nicht so schwierig, denn durch eine ausgeklügelte Adressierung der Pixel ist für diesen Fall vorgesorgt.
Der Adapter braucht außer einem anfänglichen Bildschirmlöschen keinerlei Initialisierung oder laufende Bedienung durch die CPU. Er arbeitet völlig autonom und bremst die CPU nicht durch irgendwelche Busrequests und dergleichen. Lediglich dann, wenn die CPU auf den Bildwiederholspeicher zugreift, wird sie um durchschnittlich 2 CPU-Takte mittels WAIT-Signal verzögert. Das entspricht dem Zeitbedarf eines halben NOP-Befehls und dient der Synchronisation der drei Zugriffsarten auf den Bildwiederholspeicher (Attribute lesen und Pixel lesen durch die Displaysteuerung sowie CPU-Zugriffe). Dadurch wird vermieden, daß CPU-Zugriffe sichtbare Störungen auf dem Bildschirm erzeugen. Diese „Schneeverhinderungs"-Schaltung ist auch der Grund dafür, daß der übrige Rechner mit dem aus dem Pixeltakt von 7 MHz abgeleiteten Systemtakt arbeiten muß.
Da die Pixel byteweise im Bildwiederholspeicher stehen, muß die Displaysteuerung im Zyklus von 8 Pixeln sowohl die Attribute als auch die Pixel lesen. Dieser Zyklus besteht aus 8 Pixelzeiten (d.h. 8 elementare Zeitabschnitte von je 143 ns, in denen jeweils ein Pixel auf den Bildschirm geschrieben wird) und ist in drei Abschnitte eingeteilt. Die Zeit von Pixel 1 bis Pixel 4 steht für CPU-Zugriffe zur Verfügung. Während der Pixelzeiten 5 und 6 werden die Attribute gelesen und in einen Zwischenspeicher übernommen, und während der Pixelzeiten 7 und 8 werden die Pixel gelesen und mit dem Ende des Zyklus in ein Schieberegister übernommen, wo sie mit den nächsten 8 Takten seriell herausgeschoben werden. Mit der Übernahme der Pixel in das Schieberegister werden auch die Attribute vom Zwischenspeicher in das eigentliche Attributregister geladen. Versucht die CPU zu irgendeinem Zeitpunkt, auf den Bildwiederholspeicher zuzugreifen, so wird sie erst einmal mit WAIT angehalten. Am Beginn eines neuen Displayzyklus schaltet die Displaysteuerung das WAIT-Signal wieder ab, und die CPU kann nun während der folgenden 4 Pixelzeiten (gleich 2 Systemtakten) ihren Zugriff ausführen.
192 mal 256 Pixel, das sind ganau 6 KByte Bildwiederholspeicher, der im Hauptspeichervolumen des Rechners angeordnet ist. Um den Aufwand gering zu halten, wurde jeweils ein Farbattribut-Byte einem Quadrat von 8 mal 8 Pixeln zugeordnet. Das gesamte Bildfeld ist in 768 solche Quadrate eingeteilt (32 horizontal, 24 vertikal). Im Hauptspeichervolumen des Rechners ist direkt nach dem 6 KByte umfassenden Pixelspeicher der 768 Byte umfassende Farbattribut-Speicher angeordnet. Beide können zusammen durch Lötbrücken auf der Leiterplatte in 8-K-Schritten im Hauptspeicher verschoben werden. Die Vorzugsadresse ist 4000H. Sie ist durch einen Leiterzug vorverdrahtet. In diesem Fall liegt der Speicherbereich für die Pixel im Bereich von 4000H bis 57FFH und für die Attribute von 5800H bis 5AFFH. Wie die Farbinformationen im Attribut-Byte codiert sind, zeigt Tafel 1. Für je ein Quadrat von 8 mal 8 Pixeln kann eine von 8 Hintergrundfarben, eine von 16 Vordergrundfarben (8 Farben in 2 Helligkeitsstufen) und Blinken/Nichtblinken gewählt werden.
Die Attribut-Bytes stehen im Speicher zeilenweise in natürlicher Reihenfolge, also zum Beispiel:

1. Zeile: 5800H ... 581FH
(5800H = links oben,
581FH = rechts oben)



BEI MONTAGE IN 95mm BAUHÖHE C4, C5 UND CS2* LIEGEND MONTIEREN

✖ ≙ DURCHKONTAKTIERUNG (NUR BEI NDKL)

○ ≙ WICKELSTIFT (FUER ADRESSE 4000H NICHT NOTWENDIG)

***Bild 1* Bestückungsplan**

Tafel 1 Inhalt der Farbattribut-Bytes

Bit	7	6	5	4	3	2	1	0
	blk	xth	grün	rot	blau	grün	rot	blau
			Hintergrundfarbe			Vordergrundfarbe		

(blk = Blinkbit, xth = extrahell)

2. Zeile: 5820H...583FH
usw. bis
24. Zeile 5AE0H...5AFFH

Im Gegensatz zu dieser einfachen Zuordnung ist die Adressierung der Pixel etwas komplizierter. Angenommen, man beschreibt die Lage der Pixel durch Koordinaten auf dem Bildschirm so, daß der Nullpunkt sich links oben befindet, die Spaltenadressen sich nach rechts und die Zeilenadressen sich nach unten erstrecken. Für die Zeilenadresse werden 8 Adreßbits benötigt, wobei sich die Adresse von 0 bis 0BFH erstrecken kann. Für die Spaltenadressen werden 5 Adreßbits benötigt, da die Stellung eines Pixels im Byte den drei niederwertigsten Bits entspricht.

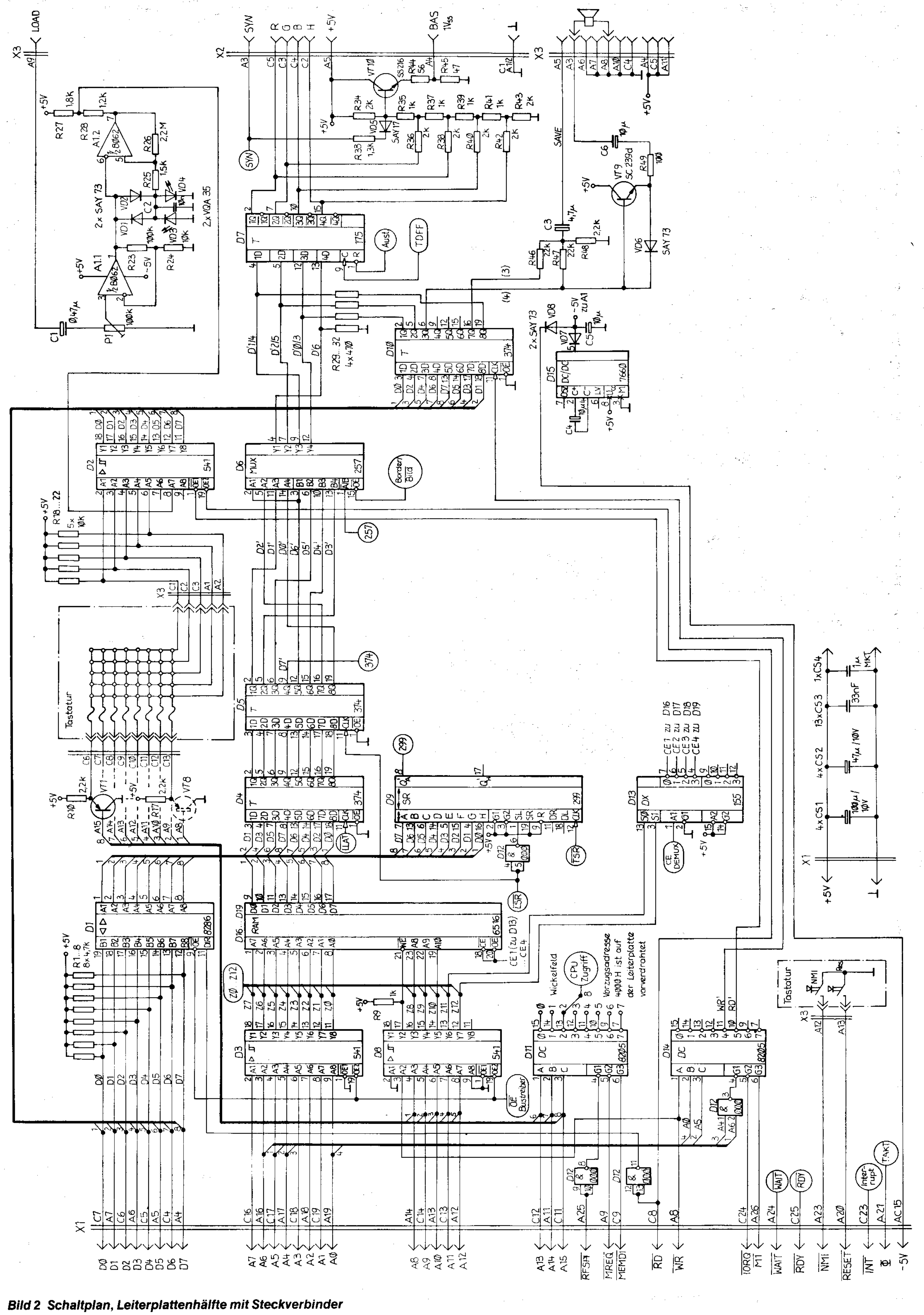


Bild 2 Schaltplan, Leiterplattenhälfte mit Steckverbinder

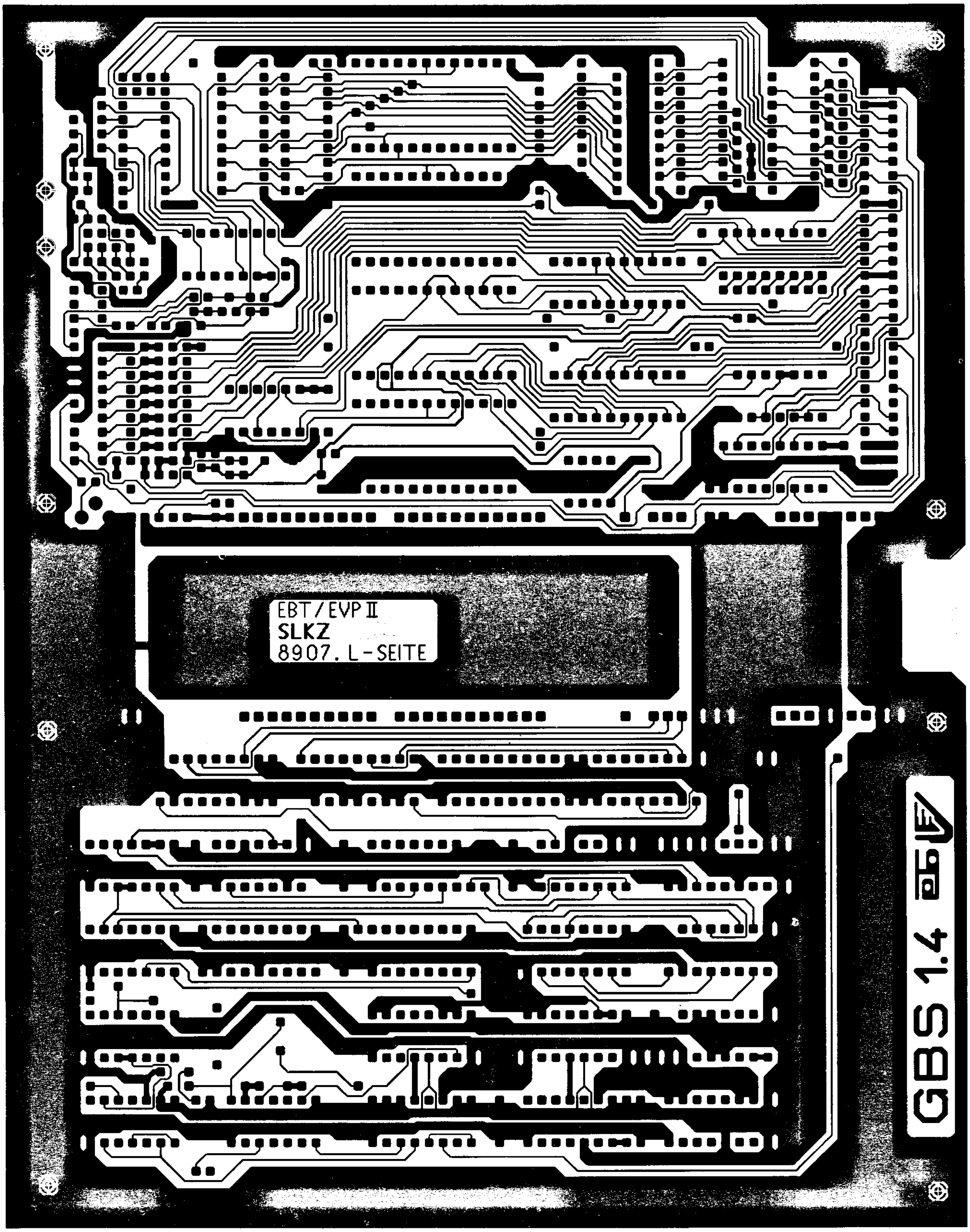


Bild 3 Layout, Lötseite

Das höchstwertige Bit im Byte gelangt zuerst auf den Bildschirm und steht demzufolge am weitesten links. Wie die verschiedenen Bits der Zeilen- und Spaltenadressen aus Sicht der CPU angeordnet sind, zeigt Tafel 2. Wie man sehen kann, sind die Zeilenadreßbits R0..R7 ineinander verschachtelt. Bei den Attributadressen sind einfach die Bits R0, R1 und R2 aus der Zeilenadresse herausgeschnitten. Auf diese Weise ist es sehr einfach, das für ein Pixel zuständige Attributbyte zu finden. Folgende kurze Programmsequenz erledigt dies:

Tafel 2 Adreßzuordnung der Pixel aus CPU-Sicht

Adreßbit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PADR	0	1	0	R7	R6	R2	R1	R0	R5	R4	R3	C7	C6	C5	C4	C3
ATADR	0	1	0	1	1	0	R7	R6	R5	R4	R3	C7	C6	C5	C4	C3

Adreßbit = Speicheradresse = Adreßbit 0 ... 15 der CPU (bei Adressierung ab 4000H)
PADR = Pixeladressen R0 ... R7 = Zeilenadresse 0 ... 191
C0 ... C7 = Spaltenadresse 0 ... 255
(Die Spaltenadreßbits C0 ... C2 entsprechen der Stellung des zum Pixel gehörigen Bits im Byte, welches an der durch C3..C7 und R0..R7 bestimmten Speicheradresse steht.)
ATADR = Attribut-Adresse, korrespondierend zu PADR

(angenommen, HL enthält die Pixeladresse nach Bild 2)

```
LD    A,H  ;Pixeladreßteil high laden
RRCA       ;um 3 Bit nach rechts
RRCA       ;verschieben
RRCA
AND   3    ;Grundadresse wegschneiden
OR    58H  ;Anfang Attributadresse dazu
LD    H,A  ;Adresse wieder zurück in HL
```

Wenn man sich darauf beschränkt, Textzei-

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
ISSN 0232–2892 · Mikroprozessortechnik · Berlin · 3 (1989) · S. 1–384

Jahresinhaltsverzeichnis 1989
VEB Verlag Technik Berlin

Mikroprozessortechnik

Übersichtsbeiträge

	Heft/Seite
Künstliche Intelligenz in Gegenwart und Zukunft *Posthoff, Ch.; Staudte, R.*	1/3
Innovative Computerarchitektur – Transputer *Schlechter, J.; Dabbagh, S.*	1/11
Einsatz von Transputern in Spracherkennung und Robotik *Neumerkel, D.; Naunin, D.*	1/13
Wegbereiter der Informatik – Brook Taylor *Biener, K.*	1/14
Objektorientierung in der Informatik *Kühle, F.*	2/41
Wegbereiter der Informatik – Johann Carl Friedrich Gauß *Biener, K.*	3/78
MSX – ein unbekannter Standard? *Lennartz, M.*	3/86
Hochleistungsrechner mit Mikroprozessoren *Heuer, H.*	4/99
Das schnelle 16-Bit-Mikroprozessorsystem U 80600 *Mandler, D.; Berndt, H.*	5/130
Wegbereiter der Informatik – Charles Babbage *Biener, K.*	5/155
Wegbereiter der Informatik – Philipp Matthäus Hahn *Biener, K.*	6/182
Desktop Publishing *Simon, J.*	7/199
Wegbereiter der Informatik – Pafnuti Lwowitsch Tschebyschew *Biener, K.*	7/210
Diskrete Fouriertransformation und digitale Signalverarbeitung *Förster, J.*	8/243
Digitale Meßumformer in der Verfahrenstechnik *Engel, H. O.*	9/268
Wegbereiter der Informatik – Georg Boole *Biener, K.*	9/275
Der Megabitspeicher und die Mikroprozessortechnik *Junghans, B.*	10/291
Hybridschaltkreise aus Hermsdorf *Racurow, B.*	10/298
Kontamination oder Reicht das Staubwischen in der Halbleiterfertigung? *Hillig, J. R.*	10/299
Der Apple Macintosh *Dolleschel, H.*	10/308
Wegbereiter der Informatik – Ludwig Eduard Boltzmann *Biener, K.*	10/313
Wegbereiter der Informatik – Hermann Hollerith *Biener, K.*	12/364

Schaltkreisentwurf und -herstellung

	Heft/Seite
Gate-Array und Standardzelle – dominierende Vertreter innerhalb der ASICs *Müller, D.*	1/21
Entwurf von Gate-Array-Schaltkreisen (Teil 1) *Müller, D.*	3/83
Entwurf von Gate-Array-Schaltkreisen (Teil 2) *Müller, D.; Fügert, E.*	6/168
Entwurf von Gate-Array-Schaltkreisen (Teil 3) *Müller, D.; Pauliuk, J.*	8/233
Entwurf von Gate-Array-Schaltkreisen (Teil 4) *Müller, D.*	12/365

Mikroprozessorsysteme und andere Bauelemente

	Heft/Seite
Anwendung des FDC U 8272 D *Böhl, E.*	2/39
16-Bit-Mikroprozessor U 80601 *Fritzsche, A.*	5/132
Der Buscontroller U 80606 DC *Heilbock, V.*	5/136
Schaltkreis zur Fehlererkennung und -korrektur U 80608 *Franke, R.; Meinecke, F.*	5/138
DRAM-Controller U 80610 *Schmidt, W.*	5/141
Peripherieschaltkreise CIO und SCC *Schmidt, M.*	5/147
Der Megabitspeicher U 61000 *Knobloch, J.; Scade, A.*	10/292
8-Bit-CMOS-Mikroprozessorsystem U 84C00 *Ritter, M.*	10/296

Mikrorechnersysteme einschließlich -peripherie

	Heft/Seite
Experimentalsystem für Mini-MAP *Prager, M; Schumacher, H.*	1/7
Signalanalyse und Echtzeitverarbeitung mit digitalen Signalprozessoren *Förster, J.; Scholz, D.*	1/19
EPROM-Programmiereinrichtungen aus dem IfAM Erfurt *Wrenzitzki, J.*	3/85
Dynamischer Dual-Port-RAM koppelt Mikrorechner *Thomä, E.; Däne, B.; Fengler, W.*	4/102
Logikanalyse des Mikroprozessors 8086/88 *Piepiorra, F.; Rothmann, G.*	7/202
Kopplung von UNIX-Rechnern *Giert, D.; Inhoff, U.*	8/227
Typenraddrucker mit IFSS an V.24-Schnittstelle *Kramer, M.*	9/268
Arithmetik für die Meßtechnik *Wand, H.*	10/312
Anwendung einer programmierbaren Kommunikationskarte *Ludwig, Ch.; Heckert, R.*	11/331
Multiprozessorsysteme mit U 8000 *Weicker, F.; Hamann, J.*	11/333
Eingabe paralleler Prozeßdaten *Friedemann, R.; Johne, E.; Röller, F.*	12/358

Software

	Heft/Seite
Programmieren in Prolog *Killenberg, H.; Knauf, R.*	1/5
Diskettenformate CP/M-kompatibler Betriebssysteme *Dames, W.*	2/35
dBase IV *Fischer, M.*	2/44
8 × 8 – ein Font für alle Fälle *Bauer, T.*	2/53
dBase III-Referenzkarte *Grafik, W.*	3/2. US
A 7100-Grafik *Krannich, K.-D.*	3/67
Grafikbibliotheken für A 7100/7150 *Stephan, K.; Thalmann, D.*	3/70
Grafikprogrammierung mit SCP-GX *Vogel, M.; Eger, A.*	3/71
Erfahrungen beim Übergang von dBase II zu dBase III *Grafik, W.*	3/72
Automatische Generierung von BASIC-INLINE-POKEs *Hanisch, Ch.*	3/75
Indizierte Variablen und Felder unter REDABAS *Streubel, T.*	3/88
Nutzereigene Laufzeitfehlerbehandlung bei mathematischer Software *Weber, K.-D.*	4/105
Basic-Dateien unter Turbo-Pascal *Gebhardt, R.; Eger, K.-H.*	4/107
Environment-Support für Turbo-Pascal *Hanisch, Ch.*	4/117
Fortschritte in dBase III Plus *Grafik, W.*	4/118
dBase III Plus-Referenzkarte *Grafik, W.*	4/3. US
Turbo-Pascal und Filesharing *Finze, W.*	6/163
Floppy-Treiber mit FDC U 8272 im Nicht-DMA-Modus (Teil 1) *Löber, Ch.*	6/166
Tools zur BATCH-File-Programmierung *Hanisch, Ch.*	6/171
Softwareansteuerung von Schrittmotoren *Grunewald, U.*	6/173
Mittel und Methoden der Künstlichen Intelligenz (Teil 1) *Schmidt, J.*	6/179
Wie funktioniert verteilte Arithmetik? *Rieken, R.*	6/181
Diskettenkapazität unter SCP 1700 mit Turbo-Pascal *Babinsky, W.*	6/181
Logischer Entwurf von Datenbanken *Zeier, T.*	7/195
Floppy-Trieber mit FDC U 8272 im Nicht-DMA-Modus (Teil 2) *Löber, Ch.*	7/196
Umleitung für SUBMIT ist schneller *Isekeit, F.*	7/205
Externe Unterprogramme in Turbo-Pascal *Hanisch, Ch.*	7/206
Entwicklung zuverlässiger Software für die Gerätetechnik *Schneider, U.*	7/212
Portabel programmieren mit C *Hübner, U.*	8/229
Fallen in C *Lau, B.*	8/231
Maus-Anwendung unter Turbo-Pascal *Schimpf, B.*	8/232
Fraktale in Pascal *Biener, B.*	8/236
Bildschirmsteuerung unter Turbo-Pascal 4.0 und DCP *Zschockelt, P.*	8/238
Interaktive Datenanalyse und dynamische Grafik *Nagel, M.; Huber, T.*	9/259
Verwaltung programmierbarer Zeichengeneratoren *Hirschmann, R.*	9/261
Echtzeituhr für Mikrorechner *Malsch, K.*	9/263
Heap-Nutzung unter Turbo-Pascal auf 16-Bit-Rechnern *Matzke, B.*	9/265
Interrupt und Einzelschritt in Forth *Taege, P.*	9/275
Mittel und Methoden der Künstlichen Intelligenz (Teil 2) *Schmidt, J.*	9/276
Erfahrungen mit Modula-2-Compilern *Lampe, J.*	10/301
Command Master *Holländer, A.*	10/307
Von Turbo 3.0 zu Turbo 5.0 Bauer, T.	11/324
Druckerinitialisierung *Matzke, B.*	11/327
DOS-Gerätetreiber für SCOM-LAN *Schulze, U.*	11/328
Portabler Bildschirmzugriff in Unix *Einert, E.; Luthe, U.*	11/340
MES – Ein dialogfreundliches Echtzeitbetriebssystem *Bonitz, F.; Neunast, J.; Rockmann, A.; Rudorfer, J.*	11/342
16-Bit-Betriebssysteme der Echtzeitverarbeitung *Hardt, W.-D.*	12/355
Nutzerspezifische Gerätetreiber unter MS-DOS *Reinhardt, M.*	12/359
Hardwarenahe Programmierung in C *Schneider, U.; Bauer, D.*	12/361
Große Programmsysteme in C auf 8-Bit-Rechnern *Götze, B.; Ambos, F.*	12/363
Überlagerung von Bildschirmfenstern *Liske, R.*	12/371
Mittel und Methoden der Künstlichen Intelligenz (Teil 3) *Schmidt, J*	12/372

Heft/Seite

MP-Kurs

Mikroprozessorsystem K 1810 WM86 (Teil 6) 2/45
Münzer, B.-G.; Jorke, G.; Engemann, E.; Kabatzke, W.; Kamrad, F.; Schumacher, H.; Stachowiak, T.
Einführung in Forth-83 (Teil 1) 3/79
Pfüller, H.; Drewelow, W.; Lampe, B.; Neuthe, R.; Woitzel, E.
Mikroprozessorsystem K 1810 WM68 (Teil 7) 4/109
Münzer, B.-G.; Jorke, G.; Engemann, E.; Kabatzke, W.; Kamrad, F.; Schumacher, H.; Stachowiak, T.
Einführung in Forth-83 (Teil 2) 5/143
Pfüller, H.; Drewelow, W.; Lampe, B.; Neuthe, R.; Woitzel, E.
Turbo-Pascal-Praxis (Teil 1) 6/175
Zander, M.
Einführung in Forth-83 (Teil 3) 7/207
Pfüller, H.; Drewelow, W.; Lampe, B.; Neuthe, R.; Woitzel, E.
Turbo-Pascal-Praxis (Teil 2) 8/239
Zander, M.
Einführung in Forth-83 (Teil 4) 9/271
Pfüller, H.; Drewelow, W.; Lampe, B.; Neuthe, R.; Woitzel, E.
Turbo-Pascal-Praxis (Teil 3) 10/303
Zander, M.
Einführung in Forth-83 (Teil 5) 11/335
Pfüller, H.; Drewelow, W.; Lampe, B.; Neuthe, R.; Woitzel, E.
Turbo-Pascal-Praxis (Teil 4) 12/367
Zander, M.

Tips & Tricks

Kurzes Maschinenprogramm mit großem Effekt 1/25
Boltze, L.
Rechnen mit komplexen Zahlen in BASIC 1/26
Krohs, W.-D.
Ziffernprüfung 1/26
Steffens, T.
SCP-Disketten-Directory als REDABAS-Datei 2/54
Matzke, B.
RAM-Disk für den KC 87 2/55
Nestler, W.
REDABAS-Tip – e-Funktion 2/55
Steffens, T.
Zugriffsanzeige für RAM-Floppy 3/86
Brosig, R.
Basic-Run im OS für KC 87-Familie 3/86
Boltze, L.
REDABAS-Tip – Sinusfunktion 3/87
Steffens, T.
FBAS-Anschluß für Farbfernsehgeräte der Serie 3000 3/87
Salzmann, G.
Kopieren und Retten von Bildschirminhalten 3/87
Schlittermann, H.
64-KByte-RAM-Module als Datenspeicher für Basic-Programme im KC 85/3 4/120
Jagdmann, E.; Domschke, W.
Abwechslung im Menü 4/121
Obenaus, G.
Prozentuale Parameter in Turbo-Pascal 5/152
Hanisch, Ch.
Die Nutzung der Routinen des Basic-Interpreters 5/153
Zierott, A.
Turbo-Pascal-Routine für den A 7100 5/155
Matzke, B.
Commodore-Interface am U 800 6/183
Seidenspinner, R.-H.
Bandlisten beim KC 87 6/184
Wobst, R.
Eine Bemerkung zu Copy Version 4.3 6/184
Wobst, R.
Programm Rollen rechts 6/184
Noske, T.
Abfrage des Druckerstatus am AC A 7100 7/214
Pietzsch, D.
Bildschirminhalt auf Kassette speichern 7/214
Luther, H.
Strings plotten in Großschrift 7/215
Noske, T.
Schnelle Algorithmen für Geraden und Kreise in Forth 8/246
Taege, P.
KC 85/3 als Eingabeterminal für PC 1715, BC A 5120 8/246
Klink, H.-K.
MS-DOS-Tip – Kursormanipulation 9/270
Schmidt, E.
Grafik im Hardcopy 9/280
Bauer, T.
Programm Rollen links 9/281
Noske, T.
Starten von Programmen aus Turbo-Pascal heraus 9/281
Sellmann, J.
Bildschirmansteuerung des PC 1715 10/314
Bauer, T.
Umcodieren der Steuertasten unter MS-DOS 10/314
Lennartz, M.
Komfortable formatierte Eingabe in Basic 10/315
Schenk, H.
Schnelles Löschen beliebiger Fenster für den KC 87 ... 11/344
Schöne, U.
Schnelles Löschen beliebiger Fenster für den KC 85/2 und /3 11/344
Bartheld, V.
Redabas-Tip – Zu einigen Detailproblemen 11/344
Heinze, J.
A3-Druckertreiber für den A 7100 11/345
Tschuch, G.; Nieber, H.
Zweiter Drucker am A 7100/7150 12/374
Voigt, H.
Dienstprogramme für SCP 1700 12/375
Isekeit, F.

Berichte

44. Internationale Technische Messe Plowdiw '88 1/32
31. Zentrale Messe der Meister von morgen 2/62
Kolloquium Datenerfassungstechnik 2/64
SYSTEC 88 2/64
DTP '88 2/64
DECWORLD '88 3/96
Electronica '88 4/126
6. Problemseminar Programmierung von Rechenanlagen des SKR 4/126
UNIX-Börse 4/126
Mikroelektronik '88 4/127
Kolloquium zu datenbankgestützter CAD-Software 4/127
ICCD '88 5/158
1. Fachtagung „Anwendung von 32-Bit-Rechenanlagen des SKR" 5/158
Fachtagung KomCom '89 6/190
Fachtagung Computeranimation 6/190
Hannover Messe CeBIT '89 6/191
Leipziger Frühjahrsmesse 1989 (Teil 1) 7/221
Leipziger Frühjahrsmesse 1989 (Teil 2) 8/253
Neues von Videoton 8/256
Kolloquium zu sicherheitskritischer Software an der Verkehrshochschule 9/287
Desktop Publishing und Druckerei 9/287
Fachgruppe Forth im Kulturbund 9/287
INTERDECK '89 9/288
13. Mikroelektronik-Bauelementesymposium 10/2. US
89. Budapester Internationale Messe 11/350
Leipziger Herbstmesse 1989 12/381
Atari-Messe 1989 12/384

Literatur

TURBO-PASCAL 1/27
Neue Technik – alte Gesellschaft 1/27
Das große Computer-Viren-Buch 1/27
Problemlösen mit PROLOG 1/27
MS-DOS 2/59
Schaltungssammlung für den Amateur 2/59
CP/M in der Praxis 3/92
Der UNIX-Werkzeugkasten – Programmieren mit UNIX 3/92
Maschinensprache des IBM-PC/AT 3/92
Serielle Busse: neue Technologien, Standards, Einsatzgebiete 3/93
Professor Schreiners UNIX-Sprechstunde 3/93
Wirksamer sprechen – hören – sehen 4/128
Technologien im Umbruch 4/128
MS-DOS-Handbuch 4/128
Encyclopedia of microcomputers 4/128
Neue Medien – Totale Mattscheibe und neue Informationsordnung 5/157
Die innovativen 80286/80386-Architekturen 5/157
Die intelligente Maschine 6/185
MS-DOS für Fortgeschrittene 6/185
UNIX 6/185
FORTH 6/185
Computerwissen für alle 7/220
MODULA-2 8/252
TECHNIK EN MINIATURE 8/252
dBase III Plus in 100 Beispielen 8/252
Künstliche Intelligenz 8/252
UNIX-Werkzeuge zur Programmentwicklung 9/286
Anwendungshilfen und Gestaltungsmuster für den optimalen Einsatz von Desktop Publishing 9/286
Mit C zum Ziel: Das Buch für Ein- und Umsteiger 9/286
Ingenieure in der DDR 10/320
Computertechnik 10/320
Arithmetische Algorithmen der Mikrorechentechnik 10/320
Der 16-Bit-Mikroprozessor des ESER-PC 11/352
„Neue Medien" – Kommunikation und Information 11/352
FORTRAN 11/352
CAD/CAM – Leitungsaufgaben und Erfahrungen 11/352
Digitale Automaten 12/380
DOS kompakt und komplett 12/380
Wörterbuch der Datenkommunikation 12/380
Schriftenreihe Beiträge zur Arbeitsumweltgestaltung 12/380

Börse

Koppelprogramm PRG 600 – PC 600 zur Prozeßüberwachung 1/28
Multitaskerweiterung rMTS für CP/M 86 1/28
V.24-IFSS-Umsetzer 1/28
Die Sprachwerkzeuge YACC und LEX unter DCP und MS-DOS 1/28
ERIKCOPY 1/28
Modula-2-Entwicklungssysteme und portable Bibliothek 1/28
Erstellung des betrieblichen Kennziffernspiegels 1/29
Parallelverarbeitung unter CP/M und Turbo-Pascal 1/29
Universelle K 1520-Schnittstellensteuerung für Drucker und Plotter 1/29
TMAS/GETVAR für Turbo-Pascal 1/29
Turbo-Window-Box für SCP 1700 1/29
Fakten/Recherchesystem 1/29
Leistungssteigerung des P 8000 durch verbesserte WDC-Firmware 2/57
Entwicklungshilfesystem HILFEn für LC-80 2/57
Hard- und Software für den KC 87 2/57
KC 85/3 im V.24-Verbund für Computerkabinette 2/58
TDABA 2/58
Hard- und Softwarekomponenten für LOTUNET 2/58
Abrechnungssystem für DCP 2/58
Nutzung von dBase-Indexdateien unter Turbo-Pascal 2/58
Umwandeln von SCP-Dateien in DCP 1700-Dateien 2/58
Ausdruck von Zeichnungen auf Nadeldruckern 2/58
Bildschirmorientierter Editor BiTEX für P8000 2/58
Drucker, Farbtücher, Reinigungsdiskette 3/90
TURBO-DATA 3/90
Textverarbeitung und Kassettensteuerung für KC 87 3/90
Kopplung ESER – A 7100 3/90
Grafischer Editor CAD88 3/90
Druckerprogramm für A 7100 3/90
FastSearch 3/90
Programmgenerator HSWGEN1 3/91
Stücklistenprozessor auf KC 85/3 3/91
Datenerfassungs- und Statistikprogramm 3/91
Hardware-RESET für Schneider-PC und Software für Rechnerkommunikation 3/91
SD 1154 an IBM-kompatiblen Computern 3/91
Konvertierung MEOS – SCP 4/122
Grafikleiterkarte mit K 1520-Format 4/122
Datenbankbetriebssystem RELA 1.4/T 4/122
Softwareansteuerung von Druckern unter DCP 1700 4/122
Kopplung von V.24 und seriellem IEC-Bus über Multiplexer 4/122
Programmverwaltung auf KC 85/3 4/122
Hexadezimaleditor 4/122
GEM-Interface für Turbo-Pascal 4.0 4/122
Textbearbeitungssystem SALOTEX '87 für ATARI 800 XL und KC 85/3 4/123
Dienstprogramme für UDOS 4/123
dBase III-Dateizugriff und Nutzeroberfläche 4/123
KYRIBASE: dBase versteht Russisch 4/123
Update-Kopierprogramm UDC 4/123
Druckerprogramm WSPrint 4/123
Softwareentwicklungsarbeitsplatz SEAP 16 Version 2.0 5/156
Programmdokumentation in Turbo-Pascal 5/156
EPROMer für Kompatible 5/156
dBase II-POWER 5/156
Maus-Programmierung und Grafik am PC 1512 5/156
Ergänzungssoftware für SCP-Systeme 5/156
80 Zeichen pro Zeile für KC 85/3 5/156
Simulator für Einchipmikrorechner 6/186
Softwareprojekt für Verwaltungsarbeit 6/186
Lohn-/Gehaltsprojekt und Themenabrechnung für F/E 6/186
IFFS-Druckerweiche 6/186
Symbole und Variantenkonstruktionen für PCCAD 6/186
Zeichnen im alphanumerischen Dialog 6/186
TP für elektronische Schreibmaschinen 6/186
UNIX-Maskengenerator/-prozessor 6/187
Diskettenpflege für den PC 1715 6/187
Serielle Kopplungskarte 6/187
Speichern von Bildschirmseiten 6/187
Programmiersprache FLOKC 6/187
Verwalten von Diskettenbeständen 6/187
Macroassembler für Kleincomputer 6/187
Datenübertragungstreiber COCO 7/128
Ein neues Datenbankkonzept 7/218
Schaltungsanalyse und EPROM-Programmierung 7/218
Centronics-Schnittstelle am P8000 7/218
Pseudografiksoftware unter REDABAS 7/218

	Heft/Seite
Zugriff auf dBase-Dateien unter Turbo-Pascal 4.0	7/218
Grafik-Softwarepaket	7/218
Echtzeitentwicklungssystem für Signalprozessoren	7/218
Diskettenmanipulierung mit SETDISK	7/219
Programmpaket für die Eletronikbranche	7/219
Kopplung SD 1156 – AC 7100/50	7/219
Shell-Menü-Hilfe für P 8000/WEGA	7/219
Neue RELA-Versionen 3.0 und 4.0	7/219
Kopplung von MC 80.3x und PC	7/219
DCP-Druckertreiber für SD 1152	7/219
Basic-Vokabeln über Tastatur verfügbar	7/219
MINICALC	7/219
Kopplung KC 85/3 – PC 1715	7/219
Neues OS für COMP JU+TE R	7/219
Tourenoptimierung mit BC-TOUR, Version 2.0	8/248
Forth-Spracherweiterung für Künstliche Intelligenz	8/248
Dateikonvertierung UNIX – MS-DOS	8/248
Turbo-Pascal-Routinen für dBase III-Zugriff	8/248
Interaktives Dialogsystem für P8000	8/248
Aufarbeiten von Daten auf Lochband	8/248
Software für Gaststätten	8/248
Konvertierungssystem für Leiterplattendaten	8/249
Anschluß VT 27000 an 8-Bit-Mikrorechner	8/249
Festplatte für EC 1834 und Software für Plotter	8/249
Anfertigen für Leiterplatten	8/249
Grafik-Druckprogramm für K 6313 am K 8919	8/249
Kopplung PC und SD 1156	8/249
Druckprogramm RUND	8/249
Modulbibliothek für Turbo-Pascal	8/249
MULTICAD-Kompendium	9/282
Kopplung A 7150 mit Druckern über IFSS	9/282
PENCIL und S 3004	9/282
dBase-Programm COMMWORK	9/282
Kopierprogramm COWA	9/282
MS-DOS-Grafiktoolbox unter Turbo-Pascal	9/282
TEXTPLOT	9/282
Arbeitshilfe für Turbo-Pascal 4.0/5.0	9/282
Einchiprechnersteuerungen	9/282
Editor für 8 × 8-Font	9/283
Compiler-Compiler MIRA	9/283
AUTRA-M16-Programmpakete	9/283
DISKEDIT	9/283
Magnetbandgerät CM 5300.01 über V24 an PC	9/283
Fehlersuche am EC 1055.M	9/283
Turbolader für KC 87	9/283
1 MByte Operativspeicher für A 7150	10/316
Schnittstellenanpassung setty(M) für P8000	10/316
Nutzerführung durch Funktionstastenbelegung mittes Stapelverarbeitung	10/316
Nutzerverwaltungssystem LOGON	10/316
AutoCad-Zeichnungen auf dem K 6418.02	10/316
Druckertreiber PMODE	10/316
Arbeiterbetreuung	10/316
Erika 3004 zur Computersteuerung	10/316
FEM-Postprozessor	10/317
Basismodule für Fortran 77	10/317
Handwerkersoftware und Systemsoftware	10/317
Schneider-PC am Farbfernseher	10/317
Dateiübertragungssystem zwischen KC und PC	10/317
Programmpaket für Handwerksbetriebe	10/317
Leistungsverzeichnis und Rohrnetzberechnung	10/317
Papiersparender Druck von Textdateien	10/317
Wir installieren Ihre Festplatte	11/346
Turbo-Editor unter WEGA	11/346
Dateikonvertierung am P8000	11/346
Informationssystem für den KC 85/3	11/346
Funktionstastenkodierung und schnelle Alphanumerik am A 7100	11/346
EPROM-Programmiergerät mit Schnittstelle V.24	11/346
Erzeugnisübersichten	11/346
Mathematische Grundfunktionen für REDACOM-Programme	11/347
KC 87 am ROLANET 1	11/347
Grafischer Druck von Meßreihen und Funktionen	11/347
Programme für Handwerks- und Handelsbetriebe	11/347
Kopplung A 7150 – DZT 912 RS	11/347
CP/M-Hardwareerweiterung mit zugehörigem BIOS	11/347
Zwei Routinen für KC 85/3	11/347
DCP für 16-Bit-PCs ohne Festplatte	12/376
Kopplung EMR-PC	12/376
Driver für IFFS-ZVE-Schnittstelle am A 7150	12/376
Nutzeroberfläche für 8-Bit-PCs	12/376
Mudulare Programmothek für den KC 87	12/376
Datenbankbaukasten dB-Box	12/376
Portabilität getypter Pascal-Dateien	12/376
PETRI-Netz-Interpreter für Einchipmikrorechner	12/376
Abrechnung aller Fuhrparkleistungen	12/377
Dateibestandskontrolle unter DCP	12/377
PICPLOT zum Plotten von Bildfiles	12/377
Stromlaufpläne mit PCCAD	12/377
Prüfprogramm PRUEF für die Datenerfassung	12/377
DISKETTE	12/377
SEAP 16 Version 2.1	12/377

Entwicklungen und Tendenzen

	Heft/Seite
MC 68030-Prozessor mit 33 MHz	1/30
16-Bit-Spezialist für Forth	1/30
Ventura Publisher Version 2.0	1/30
EPROMs mit 4 und 8 MBit	1/30
Schneller Transputer-Chip	1/30
Superminicomputer MV 40000	1/30
Kombination von Silizium- und Galliumarsenid-Technik	1/30
Betriebssysteme in Konkurrenz	1/30
Neuer Supraleiter	1/30
Schritte zur Realisierung neuronaler Rechnernetze	1/31
Bildverarbeitung mit Video-Geschwindigkeit	1/31
LCD-Farbbildschirm von Sharp mit 1,2 Mio Bildpunkten	1/31
Forschungen zur Molekularelektronik in Großbritannien	1/31
Weitere Kapazitätserhöhung bei 3,5-Zoll-Winchesterlaufwerken	1/31
Additivtechnik modernisiert Leiterplattenfertigung	1/31
32-Bit-Prozessor mit flexibler Architektur	2/60
KI-Workstation mit 40-MHz-Mikroprozessor	2/60
Laserdrucker mit 600 dpi	2/60
Super-VGA	2/60
Schneller Matrixdrucker von Seikosha	2/60
PS/2 in China?	2/60
32-Bit-GaAs-Mikroprozessor	2/60
16-Bit-uP bald mit 33 MHz?	2/60
Neue Macintosh-Version IIx	2/60
Toshiba entwickelt 32-Bit-Mikroprozessor für TRON-Projekt	2/61
Masscomp: Echtzeit-System auf UNIX-Basis	2/61
Löschbare Optik-Speicher	2/61
Farbe auf Weiß	2/61
Neues japanisches Computerprojekt	2/61
Weitere Ergebnisse der Mikromechanik	2/61
IBM informierte über Entwicklungsrichtungen	2/61
Diskettenspeicher großer Kapazität	2/61
Pläne für Hochleistungsrechner in Indien	2/61
Silizium für optoelektronische Bauelemente	2/61
Automatische Textübersetzung Englisch – Chinesisch	2/61
MS-DOS-Version 4.01	3/94
Leistungsstarker Universalrechner für kommerzielle Anwendungen	3/94
Neues Open Access	3/94
A0-Format-Laserplotter vorgestellt	3/94
MC 68040 angekündigt	3/94
Neue Chip-Fertigungsstraße bei IBM	3/94
RISC wird schneller	3/94
Wordstar 5.0	3/94
Neue Chips von TI	3/94
3500 PCs nach China	3/94
Projekt eines ultrakompakten Computers	3/94
Flüssigkristallanzeige	3/94
Klein und kraftvoll	3/95
Ursachen und Chipausfälle	3/95
Nach SMD kommt IMD	3/95
Verfahren zur Gewinnung von Galliumarsenid	3/95
Röntgenlithographie zur Chipherstellung	3/95
100 ns schnelle 1-MBit-EPROMs	3/95
Speicherchip mit Supraleiter	3/95
Erster optischer 1-KBit-Chip	4/124
Floptical Disk mit mehr als 20 MByte	4/124
MCI Carat IV	4/124
32-Bit-Europrozessor	4/124
Laptop mit Farbdisplay	4/124
Neue CrystalPrint-Familie von Qume	4/124
BRD-Forscher entwickelten superschnellen Siliziumchip	4/124
Künftig 32-Bit-RISC-Prozessoren von Siemens	4/124
Intel-kompatible Prozessoren von NEC	4/124
16-MBit-DRAM von Matsushita	4/125
Neue Laptop-Computerreihe von Sharp	4/125
Transistoren und Dioden aus Polyacetylen	4/125
Hochleistungschip von NEC	4/125
GaAs-Chip mit Schaltfrequenz von 3,5 ps	4/125
Weitere Rechnerprojekte mit Galliumarsenid-Schaltkreisen	4/125
Vorbehalte hinsichtlich des Nutzens von Hochtemperatursupraleitern	4/125
Projekt „5. Rechnergeneration in Japan" mit Verzögerung	4/125
Forschungen für 64-MBit-DRAM	5/159
Leiterplattenentflechten mit Transputerhilfe	5/159
Der PC-Markt '88 in den USA und in der BRD	5/159
„Raucherfreie" Halbleiterfertigung	5/159
Intel System 520	5/159
Flache Kathodenstrahlbildröhre	5/159
Bleistifte bleiben Arbeitsmittel	5/159
100 MByte auf Glas-Festplatte	5/159
Intel 80486 angekündigt	6/188
Motorola 68030 mit 50 MHz	6/188
HP-Workstation mit 68040 geplant	6/188
Neuer 16-Bit-PC von Compaq	6/188
Weitek-Koprozessor auch für 68020/68030-CPU	6/188
Computer kann 3755 chinesische Schriftzeichen erkennen	6/188
POWER-RAMs	6/188
Schneller RAM mit Pipeline-Technik	6/188
Mini-Bildschirm	6/189
Neurochips von Mitsubishi	6/189
Zweidimensionaler Draht	6/189
Einblicke in die Mega-Welt...	6/189
Prozessor mit 16.5 MIPS	7/216
CADy Junior	7/216
PostScript-Farbdrucker	7/216
Entwicklungstendenzen bei Bauelementen im kapitalistischen Markt	7/216
Erste PC-Familie mit 80486-Prozessor	8/250
Compaq Deskpro 386/33	8/259
EMS-Chip	8/250
Hochauflösender Farbscanner ACS 100 Color	8/250
Spezialprozessor für Künstliche Intelligenz	8/250
Integration bei elektronischen Bauelementen	8/250
Optisches Netz mit Kunststoffasern	8/250
Megabit-RAMs werden schneller	9/284
32-Bit-PCs von Schneider	9/284
Joint-venture zur Produktion von Halbleiterspeichern in den USA geplant	9/284
Hewlett-Packard übernimmt Apollo	9/284
Personalcomputer im Aufwind	9/284
EPROMs bis 20 MHz	9/284
Erste BiiN-Modelle	9/284
Molekulare Speicher aus organischen Molekülketten	9/285
Optische Rechner können elektronische Systeme nicht ersetzen	9/285
Glasfaser der dritten Generation	9/285
GaAs-Transistor als EEPROM-Element	9/285
Japan produziert 4-MBit-Speicher	9/285
Dreidimensionale IC-Prototypen	9/285
CrystalPrint Publisher	9/285
dBase für DEC-Rechner	9/285
Produktpalette von DEC erweitert	10/318
Neue PCs von IBM	10/318
Weitek-Koprozessor für Intel 80486	10/318
Mobile Datenbestände auf Festplatten	10/318
Zeichnungsverwaltung für CADdy	10/318
Turbo-Datenbank	10/318
Kupfer-Plast-Verbindung mit doppelter Festigkeit	10/319
Bauelemente mit Quantenstrukturen	10/319
Bondbare Schicht für einen keramischen Chip-Carrier	10/319
Bildchip mit mehr als 4 Mio Elementen	10/319
Optische Speicher mit Phasenwechsel	10/319
Vakuum-Mikroelektronik schaltet extrem schnell	10/319
Flachdisplay mit großer Helligkeit	10/319
Hardcopy-Bilder ab TV	10/319
RISC-Chip mit 33 MHz	11/348
Steigerung der Chipproduktion in Ungarn geplant	11/348
80x86-Reihe ohne Ende?	11/348
Umrüsten zum Farbdrucker	11/348
Hefen als lebende Halbleiterfabriken	11/348
Vereinfachte Benutzeroberfläche für PCs und Workstations	11/348
Hochgeschwindigkeits-Chips für RISC-Systeme	11/348
Lisp für Echtzeitapplikationen	11/348
Neues Verfahren zur Herstellung von Halbleiterschichten	11/349
Drucker mit zwei Druckköpfen	11/349
Farb-Flachbildschirm für tragbare Computer	11/349
Wettlauf bei der Produktion von Superchips	11/349
Elektronisches Speicher-Subsystem auch für Großrechner	11/349
Trends bei der Gehäusefertigung	11/349
Erste 486er Applikationen	12/378
32-Bit-Echtzeitbetriebssystem	12/378
Sharp Laserdrucker JX-9500	12/378
Bull-Gruppe und MIPS vereinbaren Kooperation	12/378
System-Software auf Compact-Disks	12/378
Wärmebeständige Leiterplatte	12/379
Disk-Beschriftung in 15 Sekunden	12/379
Trends bei Stromversorgungen	12/379
Digitaler Großflächenfarbkopierer	12/379
Weitere Leistungserhöhung durch 96-Bit-Architektur	12/379
Fernsehbilder auf dem PC	12/379
Nitridfilm für Magnetköpfe	12/379
Gemeinschaftsentwicklung einer Arbeitsstation	12/379

Technik international/ vorgestellt

	Heft/Seite
Workstations *Horn, T.*	1/24
Laptops	1/4. US •
Compaq Deskpro 386/25	2/56
„CIM ist mehr als nur Computer"	2/4. US
PC-Hauptspeichererweiterungen	3/89
Supercomputer	4/4. US
EAW electronic P 8000 compact	5/160

Heft/Seite
Komfortable Drucker 6/2. US
Tastaturen 7/217
Intel i860 8/251
Die schnellen PCs 8/4. US
Interaktive Echtzeitgrafik 9/2. US
Der 80486 und seine Familie 9/278
RISC-Workstation jetzt auch von DEC 9/4. US
Horn, T.
Der Megabitspeicher U 61000 10/4. US
Jobs' Cube – der neue Zauberwürfel? 11/343
Tulip at 386 sx 12/4. US

Autorenverzeichnis

Heft/Seite

A
Ambos, F.
s. Götze, B.

B
Babinsky, W. 6/181
Bartheld, V. 11/344
Bauer, D.
s. Schneider, U.
Bauer, T. 2/53
9/280
10/314
11/324
Berndt, H.
s. Mandler, D.
Biener, B. 8/236
Biener, K. 1/14
3/78
5/155
6/182
7/210
9/275
10/313
12/364
Böhl, E. 2/39
Boltze, L. 1/25
3/86
Bonitz, F.; Neunast, J.; Rockmann, A.; Rudorfer, J. 11/342
Brosig, R. 3/86

D
Dabbagh, S.
s. Schlechter, J.
Dames, W. 2/35
Däne, B.
s. Thomä, E.
Dolleschel, H. 10/308
Domschke, W.
s. Jagdmann, E.
Drewelow, W.
s. Pfüller, H.

E
Eger, A.
s. Vogel, M.
Eger, K.-H.
s. Gebhardt, R.
Einert, E.; Luthe, U. 11/340
Engel, H. O. 9/268
Engemann, E.
s. Münzer, B.-G.

F
Fengler, W.
s. Thomä, E.
Finze, W. 6/163
Fischer, M. 2/44
9/287
Förster, J. 8/243
Förster, J.; Scholz, D. 1/19
Franke, R.; Meinecke, F. 5/138
Friedemann, R.; Johne, E.; Röller, F. 12/358
Fritzsche, A. 5/132
Fügert, E.
s. Müller, D.

G
Gebhardt, R.; Eger, K.-H. 4/107
Giert, D.; Inhoff, U. 8/227
Götze, B.; Ambos, F. 12/363
Grafik, W. 3/2. US
3/72
4/118
4/3. US
Grunewald, U. 6/173

H
Hamann, J
s. Weicker, F.
Hanisch, Ch. 3/75
4/117
5/152
6/171
7/206
Hardt, W.-D. 12/355
Heckert, R.
s. Ludwig, Ch.
Heilbock, V. 5/136
Heinze, J. 11/344
Heuer, H. 4/99
Hill, H.-J.
s. Hemke, H.
Hillig, J. R. 10/299
Hirschmann, R. 9/261
Holländer, A. 10/307
Horn, T. 1/24
9/4. US
Huber, T.
s. Nagel, M.
Hübner, U. 8/229

I
Inhoff, U.
s. Giert, D.
Isekeit, F. 7/205
12/375

J
Jagdmann, E.; Domschke, W. 4/120
Johne, E.
s. Friedemann, R.
Jorke, G.
s. Münzer, B.-G.
Junghans, B. 10/291

K
Kabatzke, W.
s. Münzer, B.-G.
Kamrad, F.
s. Münzer, B.-G.
Killenberg, H.; Knauf, R. 1/5
Klink, H.-K. 8/246
Knauf, R.
s. Killenberg, H.
Knobloch, J.; Scade, A. 10/292
Kramer, M. 9/268
Krannich, K.-D. 3/67
Krohs, W.-D. 1/26
Kühle, F. 2/41

L
Lampe, B.
s. Pfüller, H.
Lampe, J. 10/301
Lau, B. 8/231
Lennartz, M. 3/86
10/314
Liske, R. 12/371
Löber, Ch. 6/166
7/196
Ludwig, Ch.; Heckert, R. 11/331
Luthe, U.
s. Einert, E.

M
Malsch, K. 9/263
Mandler, D.; Berndt, H. 5/130
Matzke, B. 2/54
5/155
9/265
11/327
Meinecke, F.
s. Franke, R.
Müller, D. 1/21
3/83
12/365
Müller, D.; Fügert, E. 6/168
Müller, D.; Pauliuk, J. 8/233
Münzer, B.-G.; Jorke, G.; Engemann, E.; Kabatzke, W.; Kamrad, F.; Schumacher, H.; Stachowiak, T. 2/45
4/109

N
Nagel, M.; Huber, T. 9/259
Naunin, D.
s. Neumerkel, D.
Nestler, W. 2/55
Neumerkel, D.; Naunin, D. 1/13
Neunast, J.
s. Bonitz, F.
Neuthe, R.
s. Pfüller, H.
Nieber, H.
s. Tschuch, G.
Noske, T. 6/184
7/215
9/281

O
Obenaus, G. 4/121

P
Pauliuk, J.
s. Müller, D.
Pfüller, H.; Drewelow, W.; Lampe, B.; Neuthe, R.; Woitzel, E. 3/79
5/143
7/207
9/271
11/335
Piepiorra, F.; Rothmann, G. 7/202
Pietzsch, D. 7/214
Posthoff, Ch.; Staudte, R. 1/3
Prager, M.; Schumacher, H. 1/7

R
Racurow, B. 10/298
Reinhardt, M. 12/359
Rieken, R. 6/181
Ritter, M. 10/296
Rockmann, A.
s. Bonitz, F.
Röller, F.
s. Friedemann, R.
Rothmann, G.
s. Piepiorra, F.
Rudorfer, J.
s. Bonitz, F.

S
Salzmann, G. 3/87
Scade, A.
s. Knobloch, J.
Schenk, H. 10/315
Schimpf, B. 8/232
Schlechter, J.; Dabbagh, S. 1/11
Schlittermann, H. 3/87
Schmidt, E. 9/270
Schmidt, J. 6/179
9/276
12/372
Schmidt, M. 5/147
Schmidt, W. 5/141
Schneider, U. 7/212
Schneider, U.; Bauer, D. 12/361
Scholz, D.
s. Förster, J.
Schöne, U. 11/344
Schulze, U. 11/328
Schumacher, H.
s. Prager, M.
Schumacher, H.
s. Münzer, B.-G.
Seidenspinner, R.-H. 6/183
Sellmann, J. 9/281
Simon, J. 7/199
Stachowiak, T.
s. Münzer, B.-G.
Staudte, R.
s. Posthoff, Ch.
Steffens, T. 1/26
2/55
3/87
Stephan, K.; Thalmann, D. 3/70
Streubel, T. 3/88

T
Taege, P. 8/246
9/275
Thalmann, D.
s. Stephan, K.
Thomä, E.; Däne, B.; Fengler, W. 4/102
Tschuch, G.; Nieber, H. 11/345

V
Vogel, M.; Eger, A. 3/71
Voigt, H. 12/374

W
Wand, H. 10/312
Weber, K.-D. 4/105
Weicker, F.; Hamann, J. 11/333
Wobst, R. 6/184
6/184
Woitzel, E.
s. Pfüller, H.
Wrenzitzki, J. 3/85

Z
Zander, M. 6/175
8/239
10/303
12/367
Zeier, T. 7/195
Zierott, A. 5/153
Zschockelt, P. 8/238

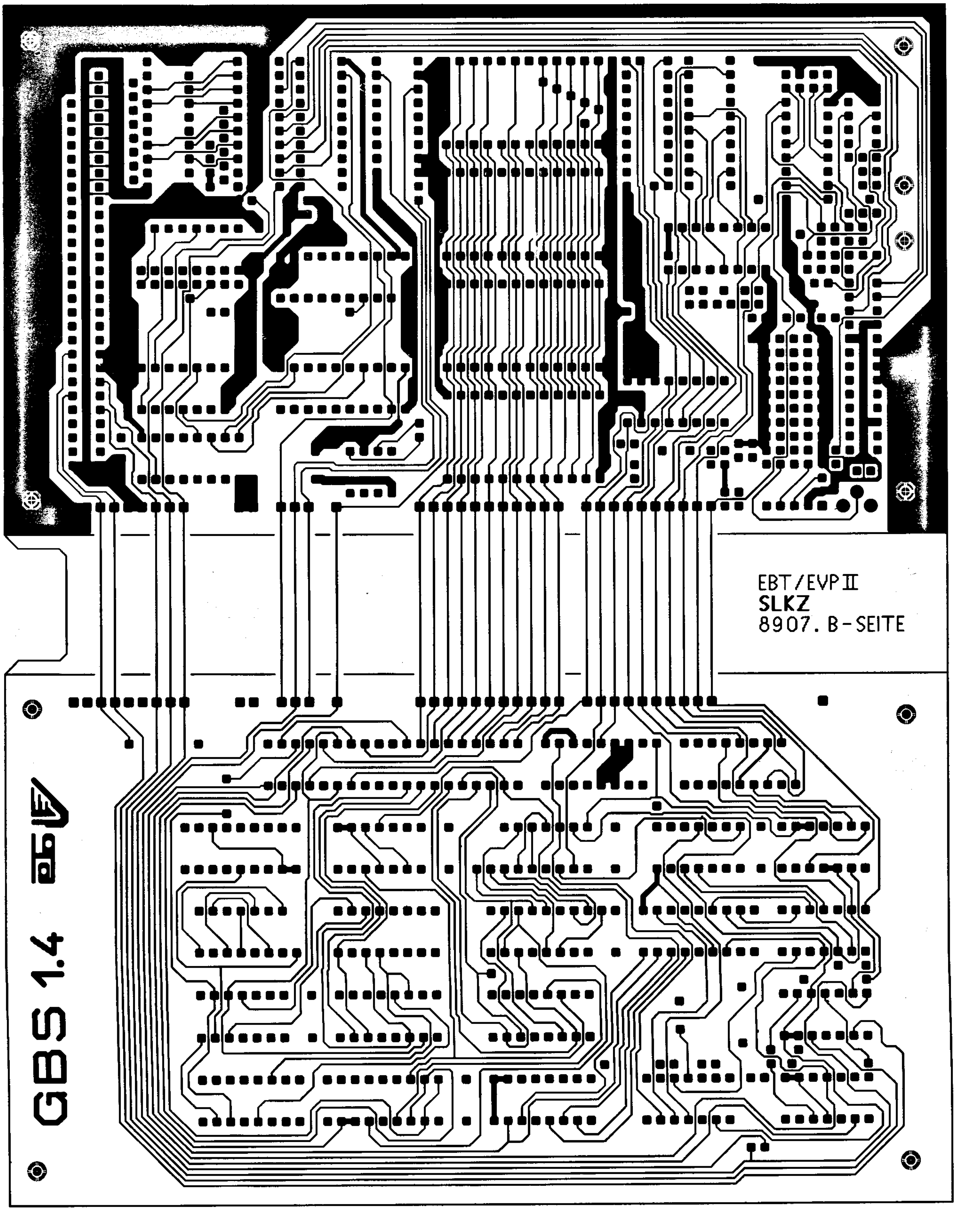


Bild 4 Layout, Bestückungsseite

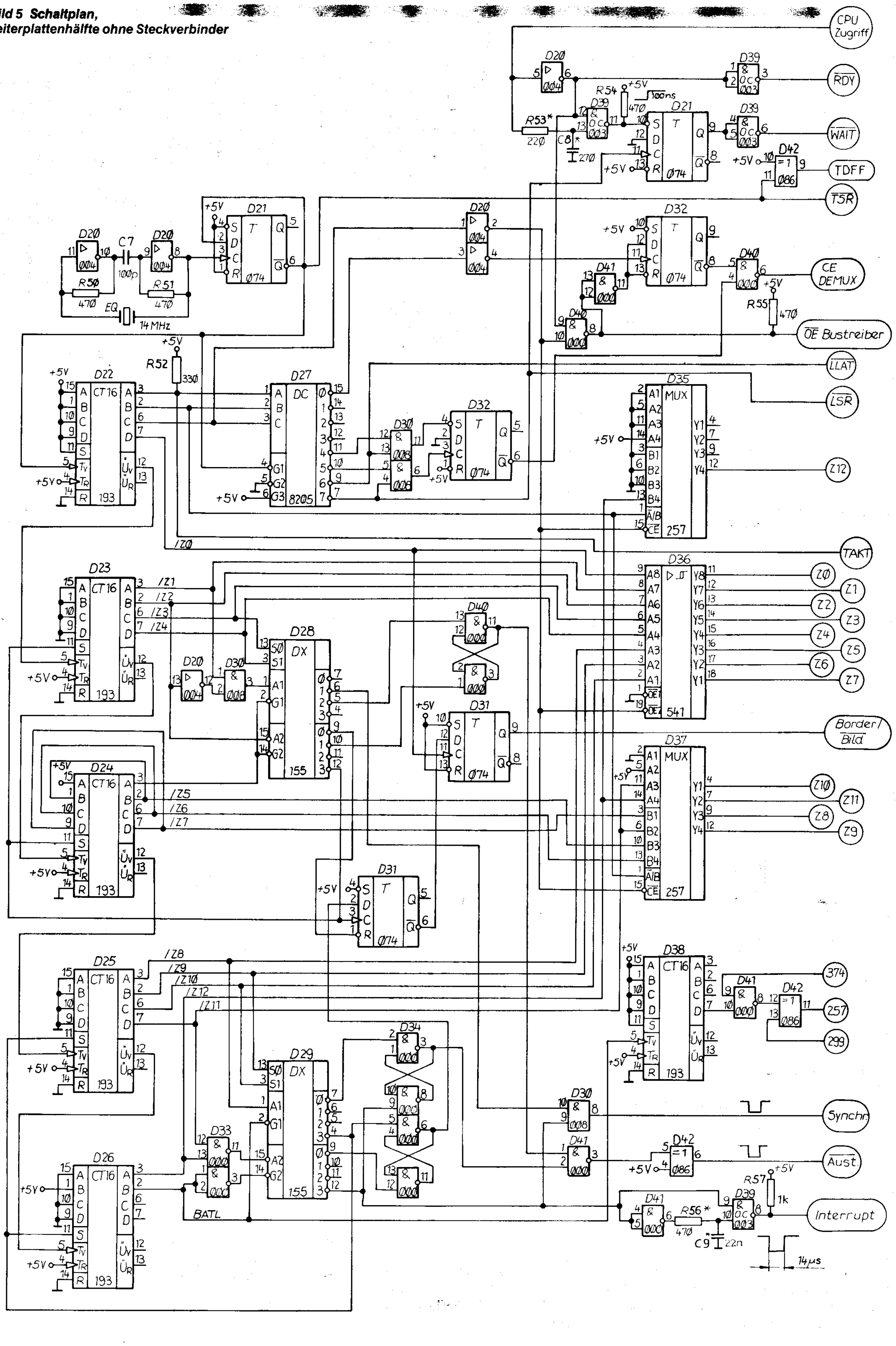
chen nicht an beliebige Stellen, sondern nur so auf den Bildschirm zu schreiben, daß ein Zeichen genau in eines der 768 Quadrate gelangt, dann ist es dank der Verschachtelung der Zeilenadreßbits auch recht leicht, Texte auf diesen Farbgrafikadapter zu schreiben. Angenommen, DE zeigt auf den Anfang eines der 768 Quadrate, in welches das Zeichen **F** geschrieben werden soll und HL zeigt auf den Anfang einer Bitmustertabelle für dieses Zeichen. Dann erledigt folgendes kleine Unterprogramm das Einschreiben des Bitmusters für das Zeichen **F** in den Bildwiederholspeicher:

```
WCHAR:  LD    B,8
ZYKL:   LD    A,(HL)    ;Pixel holen
        LD    (DE),A    ;Pixel in BWSP
                        ;schreiben
        INC   HL        ;nächstes Pixel-
                        ;byte
        INC   D         ;Pixeladresse
                        ;weiterstellen
        DJNZ  ZYKL-#    ;8mal insgesamt
        RET
```

```
;BEISPIEL FÜR BITMUSTER „F“
  DEFB 00000000B ;
  DEFB 11111110B ; *******
  DEFB 01100010B ;  **   *
  DEFB 01101000B ;  ** *
  DEFB 01111000B ;  ****
  DEFB 01101000B ;  ** *
  DEFB 01100000B ;  **
  DEFB 11110000B ; ****
```

Wie man sieht, ist diese Programmsequenz recht kurz und schnell, denn zum Weiterstellen

Bild 5 Schaltplan, Leiterplattenhälfte ohne Steckverbinder



der Pixeladressse genügt ein einfacher und schneller Inkrementierbefehl.

Üblicherweise enthalten Bildschirmadapter für richtige PCs zumindest noch einen Druckeranschluß, um den Platz, den der Adapter belegt, möglichst intensiv nutzen zu können. Auch bei dem hier vorgestellten Farbgrafikadapter findet sich etwas Derartiges. Allerdings ist es kein Druckeranschluß, sondern ein kombiniertes Bedienerinterface mit folgenden Anschlußmöglichkeiten:

- Tastaturanschluß für ein 8 × 5-Tastenfeld
- Lautsprecheranschluß für Tastenklick usw.
- Anschluß für Recorder oder optischen Lesestift.

Diese Zusatzanschlüsse belegen einige Adressen im I/O-Adreßraum des Rechners, vorzugsweise auf der Adresse 0FEH. Leider war auf der Leiterplatte nicht mehr genügend Platz, um den Port vollständig zu dekodieren. Er belegt deshalb einige Adressen mehr, als eigentlich erforderlich wäre. Die Konfiguration des I/O-Ports zeigt Tafel 3.

Zu erwähnen ist noch, daß der Farbgrafikadapter zu Beginn jedes Bildsynchronimpulses einen Interrupt erzeugen kann. Damit kann man sich in einfachen Konfigurationen auch ohne speziellen Zeitgeberschaltkreis eine Software-Uhr einrichten oder auch zyklisch die Tastatur abfragen. Allerdings kann der Adapter keinen Interruptvektor liefern, weswegen die CPU im Interruptmodus 1 betrieben werden muß. Bei größeren Systemen kann man natürlich auf diese einfache Interruptquelle verzichten, indem man R 37 entfernt und C 10 durch eine Brücke ersetzt.

Tafel 3 Zusatzport-Belegung

bei Ausgabe (OUT...)

Bit	7	6	5	4	3	2	1	0
	–	–	–	LSP	REC	grün	rot	blau
						Farbe des Randes		

(LSP = Lautsprecherausgang)
(REC = Recorderausgang)

bei Eingabe (IN...)

Bit	7	6	5	4	3	2	1	0
	–	RIN	–	SP5	SP4	SP3	SP2	SP1

(RIN = Lese-Eingang Recorder/Lesestift)
(SP1..5 = Spaltenleitungen der Tastaturmatrix)
(Die Tastaturzeilen werden über die Adreßbits A8...A15 angesteuert.)

Der Aufbau des Farbgrafikadapters

Das Layout des Adapters wurde auf dem üblichen K 1520-Leiterplattenmaß von 215 mm mal 170 mm entworfen. Viele Geräteentwickler sind allerdings seit einiger Zeit mit diesem Maß nicht mehr recht zufrieden, da es durch die übliche senkrechte Einbauweise im EGS-Gehäuse zu recht klobig wirkenden Geräten führt. Deshalb sind inzwischen auch Weiterentwicklungen des K 1520-Sortiments wie *NANOS* von der IHS Warnemünde recht beliebt geworden, die als Leiterplattengrundmaß 95 mm mal 170 mm verwenden, womit wesentlich kleinere und dem Fortschritt des Bauelementesortiments besser entsprechende Geräte herstellbar sind. Auch ein eventueller Übergang auf das Format 100 mm mal 160 mm ist damit leichter. Dieser Tendenz haben wir uns angeschlossen. Ein Blick auf das Layout des Farbgrafikadapters zeigt, daß man bei Bedarf die Leiterplatte teilen kann, womit man zwei Platten im Maß von 95 mm mal 170 mm erhält, die dann über Litze verbunden werden müssen. Damit ist der Adapter sowohl in klassischen K 1520-Systemen als auch in neueren kleinformatigen Konfigurationen verwendbar. Der Adapter benötigt als Versorgungsspannung lediglich +5 V bei etwa 630 mA, wenn man zur Versorgung des Operationsverstärkers der Recorder-/Leseschaltung einen DC-DC-Wandler U 7660 DC einsetzt. Wenn man die Leseschaltung nicht benötigt oder bereits −5 Volt im Rechner zur Verfügung hat, kann man den DC-DC-Wandler natürlich auch weglassen.

Auf eine ausführliche Schaltungsbeschreibung soll hier verzichtet werden, da bekanntlich ein Stromlaufplan mehr sagt als tausend Worte. Für die TTL-Schaltkreise kommen (soweit möglich) LS-Typen zum Einsatz. Die Schaltkreisbezeichnungen wurden auf die charakteristische Nummer verkürzt, das heißt, man lese 374 als DL 374, 8205 als DS 8205, 6516 als U 6516 D 15 usw. Der Einsatz von HCT-Schaltkreisen bringt bis auf den Ersatz der drei DS 8205 durch stromsparende U 74 HCT 138 kaum Vorteile. Kritische Stellen und Abgleichpunkte gibt es nicht, so daß bei sorgfältigem Aufbau der Farbgrafikadapter auf Anhieb funktionieren wird.

✉ KONTAKT ☎

VEB Elektronische Bauelemente Teltow, Bereich EV (für technische Fragen) oder EK (für sonstige Fragen), Ernst-Thälmann-Straße 10/15, Teltow, 1530; Tel. 45 34 59 (Spindler), 45 31 59 (Kammer), bei sonstigen Fragen 45 28 00 (Marciniak)

Arrangiert: Turbo-Pascal und ANSI

Christian Hanisch, Berlin

Wer kennt es nicht, das leidige Problem der Ausgabe von Zeichenketten mit Bildschirmattributen unter Turbo-Pascal. So vorteilhaft die Verwendung der ANSI-Steuersequenzen hierfür auch wäre, so unvermittelt versagt diese Methode bei der standardmäßigen Ausgabe über die vordefinierte Textdatei Output von Turbo-Pascal. Der Grund dafür ist, daß die ANSI-Sequenzen nur dann wirksam werden, wenn sie durch die Standarddienstleistungsroutinen des DOS-Interrupts 21H (zum Beispiel Funktion 09H) ausgegeben werden, was bei Turbo-Pascal standardmäßig nicht der Fall ist. Turbo-Pascal verwendet hier vielmehr eine eigene Ausgabe, die direkt auf die ROM-BIOS-Funktionen zugreift.

Nun kann man sich auf ganz unterschiedliche Art des Problems annehmen. Angefangen bei quelltextmäßig aufwendigen Routinen, die über *TEXTCOLOR* und *TEXTBACKGROUND* die Sache bewerkstelligen, bis hin zur schnellen INLINE-Routine, die den BIOS-Interrupt 10H verwendet, ist so manches denkbar und in der Literatur auch schon vorgestellt worden.

Ein ganz anderer Weg, der verblüffend einfach ist, basiert auf der Tatsache, daß bei Verwendung von Standard-OUTPUT des

```
 1: PROGRAM TANSI;    {        Turbo-Pascal Version 5.0           }
 2: USES crt;         { <--- entfällt in Version 3.xx
 3:   ------- Bildschirmattribute für Monochrom-Monitor ------ }
 4: CONST  cls  = #27'[2J';   {   Bildschirm löschen             }
           norm = #27'[0m';   {   normale Darstellung            }
 6:        high = #27'[1m';   {   intensive Darstellung          }
 7:        undl = #27'[4m';   {   unterstrichene Darstellung     }
 8:        blin = #27'[5m';   {   blinkende Darstellung          }
 9:        inv  = #27'[7m';   {   inverse Darstellung            }
10:
11: VAR ansi : text; i : INTEGER;
12:
13: BEGIN
14:   assign(ansi,'');
15: { assign(ansi,'OUT:');        in Turbo-Pascal 3.xx'          }
16:   rewrite(ansi);
17:
18:   write(ansi,norm+cls+
19:         'Bildschirm wurde '+undl+'gelöscht!'+norm);
20:   write(ansi,#27'[7;25HKursor mit ANSI positionieren');
21:   writeln(ansi,#27'[2B');
22:   writeln(ansi,inv+'══════════════════════════════════════'+
23:             '═══════════════════════════════════════');
24:   gotoxy(26,13);
25:   writeln(ansi,norm+'Text in normaler Darstellung');
26:   gotoxy(25,14);
27:   writeln(ansi,high+'Text in intensiver Darstellung'+norm);
28:   gotoxy(23,15);
29:   writeln(ansi,undl+'Text in unterstrichener Darstellung');
30:   gotoxy(24,16); writeln(ansi,blin+
31:                 'Blinkender Text und unterstrichen'+norm);
32:   gotoxy(27,17);
33:   writeln(ansi,inv+'Text in inverser Darstellung'+norm);
34:   gotoxy(18,19); writeln(
35:         'Die Standardausgabe kann ebenso benutzt werden!');
36:   window(5,21,50,25); clrscr;
37:   FOR i:=1 TO 50 DO BEGIN
38:     write(' Turbo-Standardausgabe in ein Fenster!');
39:     delay(50) END;
40:   REPEAT UNTIL keypressed;
41:   FOR i:=1 TO 50 DO BEGIN write(ansi,
42:             ' DOS-Standardausgabe in das gleiche Fenster!');
43:     delay(50); END;
44: END.
```

Bild 1
Demonstration der Nutzung der ANSI-Steuersequenzen in Turbo-Pascal

MS-DOS Betriebssystems (zum Beispiel die Funktion 09H des Interrupt 21H) die ANSI-Steuersequenzen immer wirksam werden. Um Turbo-Pascal-Programme zum Beispiel zur I/O-Redirektion zu bequemen, ist unbedingt die Verwendung von Standard-INPUT und -OUTPUT des Betriebssystems notwendig. Dies wird dadurch erreicht, daß die vordefinierten Textdateien *Input* und *Output* von Turbo-Pascal auf den MS-DOS-Standard-INPUT beziehungsweise -OUTPUT entweder mit der Kompilerdirektive {*$*G<nr>} beziehungsweise {*$*P<nr>} in der Version 3.xx oder durch
ASSIGN(<TextFileVariable>,'') bei den Versionen 4.0 und 5.0 umgeleitet werden. Eine weitere Möglichkeit besteht bei der Version 3.xx, indem über die in der DOS-Implementation von Turbo-Pascal zusätzlich vorhandenen logischen Geräte INP und OUT kommuniziert wird. Durch die Nutzung dieser I/O-Redirektionsmechanismen dürfte damit also auch den ANSI-Steuersequenzen in Turbo-Pascal ein Anwendungsfeld erschlossen sein. Es muß faktisch nur die Turbo-Pascal-Ausgabe auf den MS-DOS-Standard-OUTPUT umgeleitet werden.
Wie so oft, hat die Sache aber auch einen Haken. Und der äußert sich zum Beispiel beim Einsatz der WINDOW-Prozedur. Die Ausgabe über den Standard-OUTPUT von DOS schert sich nämlich wenig um eventuell gesetzte Fenster.
Man sollte es einmal ausprobieren. Das Experimentieren fördert hier viele interessante Details zutage.

Literatur

/1/ Lindner, F.: Filterprogramme und Pipes. Mikroprozessortechnik, Berlin 2 (1988) 9, S. 263–266
/2/ Heimsoeth Software GmbH & Co: TURBO-PASCAL-Handbuch Version 3.xx (6. Auflage). Oktober 1986
/3/ Heimsoeth Software GmbH & Co: TURBO PASCAL Version 4.0 Referenzhandbuch.

Datenbankbetriebssystem MIMER auch auf dem PC

Dr. Uwe Wloka
Medizinische Akademie „Carl Gustav Carus", Institut für Medizinische Informatik

Das Datenbankbetriebssystem MIMER wurde nicht speziell für PCs entwickelt, sondern dient der Ergänzung der MIMER-Anwendungen auf Mainframe-, Supermini- und Minicomputer. Auf allen diesen Ebenen ist ein Programm für den Export und Import von MIMER-Datenbanken vorhanden. MIMER auf PCs sollte also nicht als Konkurrenz zu dem bewährten dBase III (Plus), sondern als Ergänzung der MIMER-Produkte unter dem Portabilitätsaspekt und unter dem Aspekt weitergehender Systemleistungen gesehen werden.
MIMER kann schnell und problemlos auf dem 16-Bit-PCs installiert werden. Die Leistungsparameter gestatten eine effektive Anwendung und eine relativ universelle Einsetzbarkeit für Datenbankaufgaben der verschiedensten Sachgebiete.
Damit können zahlreiche Vorteile von MIMER auf ESER- und SKR-Computer auch auf dem 16-Bit-PC genutzt werden.
Eine klare Systemarchitektur, der modulare Aufbau und Möglichkeiten zur flexiblen Anpassung an verschiedene Nutzeranforderungen sowie weitgehende Portabilität zeichnen das Datenbankbetriebssystem MIMER aus.
Es wird in der DDR gegenwärtig auf folgenden Computern genutzt:

Tafel 1 Computer und Betriebssysteme, für die MIMER verfügbar ist

	ESER	[illegible]	[illegible]	32-Bit-Computer
Computer	EC 1035 bis EC 1057	SM4 SM 52-11 SM 1420 I 102 K 1630	EC 1834 A 7150 Schneider PCs XT-/AT-kompatible PCs	K 1840
[illegible]	MVT SVS SVM MVS	OS-RW DOS-RV MIX RSX11M MOOS (nur Single)	MS-DOS DCP	SVP VMS

Verschiedene Nutzerschnittstellen (Dialogschnittstelle, Wirtssprachenschnittstelle) kennzeichnen MIMER.
Der Vorteil der weitgehenden Portabilität ist verbunden mit einer Programmiersprachenunabhängigkeit, so daß von den auf verschiedenen Computerebenen benutzten unterschiedlichen Programmiersprachen zu den Leistungen der Wirtssprachenschnittstelle MIMER/DB zugegriffen werden kann.

Tafel 2 Programmiersprachen. die MIMER nutzen können

	ESER	32-Bit-Computer	SKR	16-Bit-PC
Programmiersprache	Fortran PL 1 ADA Pascal Cobol Lisp	Fortran C Pascal Cobol	Fortran Pascal ADA	Fortran C ADA Turbo-Pascal Cobol Basic

MIMER kann im Single- und Multi-User-Betrieb (unter MS-DOS nur Single-User) genutzt werden. Der relativ schnelle Zugriff zu den Daten bei weitgehenden Vorkehrungen für den Zugriffsschutz und die Datensicherheit gewährleisten eine effektive Arbeit.
Es besteht ein günstiges Verhältnis zwischen dem Leistungsumfang und dem Speicherbedarf.
Kurze Einarbeitungszeiten und eine hohe Praktikabilität gewährleisten eine hohe Akzeptanz des Systems.
Das Kernstück des Datenbankbetriebssystems ist der Data Base Handler **MIMER/DB**. Er erlaubt alle Datenbankfunktionen wie das Definieren von Datenbanken und Tabellen, das Manipulieren, das Recherchieren und das Kopieren der Daten, den Datenschutz

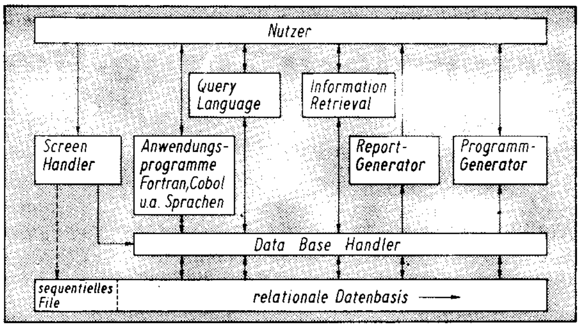


Bild 1 Komponenten des Datenbankbetriebssystems MIMER auf 16-Bit-PCs

und die Datensicherheit. Die Datenbankfunktionen werden als Routinen von MIMER/DB bereitgestellt und können von Anwenderprogrammen oder vom MIMER-System aufgerufen werden. Da die DB-Routinen auf dem PC aus Fortran aufrufbare C-Programme sind, muß die Aufrufsequenz von MS-Fortran beachtet werden.
Die DB-Routinen sind in einer Bibliothek DBLIB.LIB enthalten und werden in die Anwenderprogramme gelinkt.
MIMER/QL (Query Language) ist der Name sowohl für eine leicht erlernbare, SQL-ähnliche Definitions- und Manipulationssprache für die Datenbankarbeit im Dialog als auch für die MIMER-Komponente, die diese Datenbankarbeit ausführt.
Für häufig wiederkehrende Anfragen können QL-Kommandos und Prozedurstatements in Prozeduren zusammengefaßt, mit einem Prozedureditor bearbeitet und mit einem Prozedurhandler abgearbeitet werden.
Es besteht auch die Möglichkeit zur Abarbeitung einzelner MS-DOS-Kommandos und

zur vorübergehenden Arbeit im MS-DOS-Kommandomodus, einschließlich des Starts weiterer Programme, ohne MIMER endgültig zu verlassen.
Der Screen Handler **MIMER/SH** unterstützt das Generieren von Bildschirmmasken für die Dateneingabe (einschließlich der Datenprüfung) und für die Datenausgabe über diese Masken. Für PCs ist allerdings kein Editor verfügbar.
Ausgehend von der Definitionssprache 4QL unterstützt der Programmgenerator **MIMER/PG** die Generierung von MIMER-Anwendungsprogrammen in C. Die Programmgenerierung muß auf einem Großrechner stattfinden. Die weitere Programmentwicklung (Übersetzen, Linken) kann am PC vorgenommen werden; es ist hier nur das Laufzeitsystem verfügbar.
Der Forms Manager **MIMER/FM** dient dem Anlegen, Verändern und Benutzen von Formblättern oder Bildschirmmasken und schließt viele Leistungen von *MIMER/SH* ein. MIMER/FM bietet zwar weniger Prüfmöglichkeiten, aber eine bedeutend einfachere Entwicklungstechnologie und einen Editor.
Die Systemarchitektur der MIMER-Komponenten unter MS-DOS ist im Bild 1 dargestellt.
Zusätzlich zu den Hauptkomponenten stehen auf dem PC einige Hilfsprogramme für spezielle Aufgaben zur Verfügung.
DBADM unterstützt die Aufgaben des Datenbankadministrators beim Definieren und Löschen von Datenbanken, Zugriffsrechten und Nutzern.
DBXIM dient dem Export und Import von Datenbanken oder Tabellen zwischen verschiedenen Computern.
DBBRU fertigt Sicherheitskopien an und erlaubt das Zurückkopieren von defekten Datenbanken aus Sicherheitskopien und LOG-Datenbanken.
DBGEN generiert eine Systemdatenbank als aktives Data Dictionary von MIMER.
Zur Installation von MIMER auf 16-Bit-PCs unter MS-DOS sind die folgenden Voraussetzungen erforderlich:

Hardware
- XT-/AT-kompatibler PC
- Festplatte mit mindestens 2 MByte freiem Speicherplatz
- mindestens 512 KByte Hauptspeicher

Software
- MS-DOS ab Version 2.0
- geeignete Fortran oder C-Compiler
- 8086/8088-Objekt-Linker

In den Systemdateien CONFIG.SYS und AUTOEXEC.BAT müssen die MIMER-spezifischen Eintragungen ergänzt werden.

Literatur

/1/ Autorenkollektiv: Handbuch des Datenbankbetriebssystems MIMER (Teil 1). Schriftenreihe „Informationsverarbeitung im Hoch- und Fachschulwesen", Berlin 1986

/2/ Schreiter, D.; Wloka, U.; Dramm, P.; Rosenpflanzer, J.: Erfahrungen bei der Anwendung eines relationalen Datenbanksystems. rechentechnik/datenverarbeitung, Berlin 24 (1987) 7, S. 24

KONTAKT

Medizinische Akademie „Carl Gustav Carus", Institut für Medizinische Informatik, Fiedlerstr. 74, Dresden, 8019, Tel. 4 58 35 82

Ein verteiltes Programmentwicklungskonzept für Mikrorechner

Ralf Neuthe
Wilhelm-Pieck-Universität Rostock, Sektion Technische Elektronik

Die Programmierung von Rechnern ohne Standardperipherie oder gar von Einchipmikrorechnern stellt viele Geräteentwickler in der Prozeßautomatisierung vor nicht unerhebliche Probleme. Oft können nur Assembler oder Crossassembler (Assembler für fremde Prozessoren) eingesetzt werden. Für Testung und Inbetriebnahme der Software stehen neben den teuren und schlecht verfügbaren Emulatoren und Logikanalysatoren kaum Hilfsmittel zur Verfügung. Daraus ergibt sich eine verhältnismäßig geringe Effektivität beim Entwickeln der Programme und einen damit verbundenen erheblichen Anstieg der Softwarekosten.

In diesem Beitrag wird eine Softwarelösung vorgestellt, die ein interaktives Entwickeln und Testen von Programmen für zugeschnittene Rechnerkonfigurationen (Zielrechner) ermöglicht. Zur Programmierung, Steuerung und Beobachtung des Zielrechners braucht nur in der Entwicklungsphase ein zusätzlicher PC als Entwicklungsrechner angeschlossen zu werden. Dabei entsteht ein einfaches Mehrrechnersystem (z. B. ein intelligenter Sensor – mit eigenem Prozessor – an einem PC.

Weiterhin wird die softwareseitige Lösung der Rechnerkopplung zwischen Entwicklungs- und Zielrechner beleuchtet, da sie auch für andere Einsatzfälle von Interesse sein kann.

Zielstellungen

Es soll von dem Mehrrechnersystem sowenig wie möglich Notiz genommen werden müssen. Die Programmierung des Zielrechners soll nach den gleichen Regeln, in der gleichen Art und mit der gleichen Unterstützung möglich sein, wie die „normale" Programmierung eines PCs. Es soll der Eindruck entstehen, daß man nur an dem Zielrechner arbeitet, an den die Peripherie des Entwicklungsrechners angeschlossen wurde und auf dem die bekannten Softwarewerkzeuge des Entwicklungsrechners laufen.
Für eine möglichst gute Annäherung an diese Ziele ist die Arbeit mit der Schnittstelle für die Rechnerkopplung vollständig zu verdecken. Notwendige Kommunikationen müssen automatisch und ohne jedes Dazutun des Bedieners erfolgen.
An den Zielrechner sollten so wenig wie möglich zusätzliche Ansprüche gestellt werden. Das betrifft insbesondere den Speicherausbau. Deshalb ist die dort erforderliche Zusatzsoftware auf ein Minimum zu reduzieren. Aus diesem Grund müssen alle zusätzlich erforderlichen Aktionen für die Funktion des Gesamtsystems wie Codegenerierung und Testhilfen auf dem Entwicklungsrechner laufen. Weiterhin sollten die Forderungen an die unerläßliche Schnittstelle so gering wie möglich gehalten werden.

Das Konzept

Aufbau

Man baut ein Rechnersystem auf, bei dem sowohl Hard- als auch Software verteilt sind. Bei dieser Rechnerkopplung fungiert der Entwicklungsrechner als Master. In ihm steckt der Hauptteil der Software, die eine interaktive Kommunikation mit dem als Slave arbeitenden Zielrechner ermöglicht (Bild 1).
Die hardwareseitige Ausführung der Schnittstelle ist ohne Belang. Sie braucht nur einen bidirektionalen Datentransfer zu ermöglichen. Eine Einschränkung besteht nur in der unbedingten Forderung nach der Existenz einer für den Entwicklungsprozeß frei verfügbaren Schnittstelle. Im Bereich der Prozeßautomatisierung ist das meistens gegeben, da oft ohnehin eine bidirektionale Verbindung zur übergeordneten Rechentechnik erforderlich ist. Dort kann man eventuell sogar die Geräte dieser Ebene als Entwicklungsrechner benutzen, so daß für die Softwareentwicklung nicht einmal die Hardwaresubstanz modifiziert zu werden braucht.

Wahl der Programmiersprache

Für ein solches verteiltes System ist die Programmiersprache Forth /1/ bestens geeignet. Sie bietet gute Voraussetzungen für eine effektive Softwareentwicklung sowohl für den Entwicklungsrechner (in ihrer Eigenschaft als Hochsprache) als auch für den Zielrechner (mit ihrer Maschinennähe). Damit ergibt sich eine homogene Softwarelösung für das Gesamtsystem, da beide Rechner auf die gleiche Weise programmiert werden.
Der Betriebssystemcharakter von Forth ermöglicht die Bereitstellung der vollständigen Programmierumgebung „unter einem Dach". Das erleichtert die Simulation einer Umge-

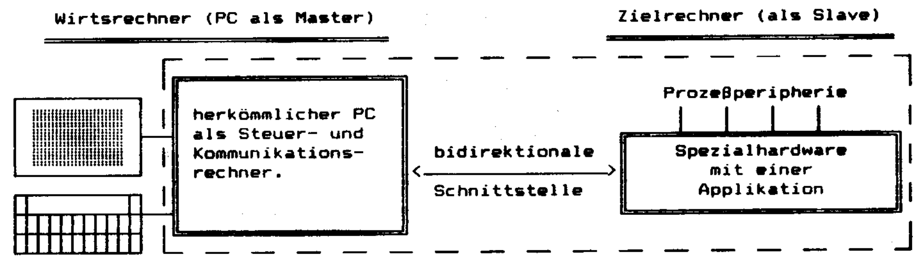


Bild 1 Das Rechnersystem

bung, in der sogar Werkzeuge für Rückübersetzung und Einzelschrittbetrieb auf dem Zielrechner laufen /2/, /3/, /4/.
Das an der WPU für die Programmierung und Inbetriebnahme von speziellen Rechnerkonfigurationen in der Automatisierungstechnik entwickelte System ist eine auf das Basissystem *comFORTH 2* nachladbare Applikation. Es trägt den Namen *comFORTH-plus*.

Bedienoberfläche

Da die gesamte Software in Forth programmiert wurde, konnten bei der Gestaltung der Bedienoberfläche die speziellen Dialog-Eigenschaften dieser Sprache (Präfix-Notation, Aufruf der Programme durch das Nennen ihres Namens) für eine Vereinfachung des Gesamtkonzepts ausgenutzt werden.
Die Arbeit mit den beiden Rechnern erfolgt nach einem sehr einfachen Prinzip: Man gibt auf dem Entwicklungsrechner durch die Wahl der Betriebsart an, mit wem gearbeitet werden soll. Dafür existieren ein Host-Modus für den Entwicklungsrechner selbst und ein Target-Modus für den Zielrechner. Entsprechend werden alle Angaben wie Wortnamen (vergleichbar mit symbolischen Programmadressen) oder Speicherzugriffe auf den jeweils ausgewählten Rechner bezogen.
Im Host-Modus verhält sich das Softwaresystem so, als ob man mit einem gewöhnlichen Forth arbeitet. Mit der Umschaltung auf den Target-Modus ändern sich für den Bediener scheinbar nur die Bezüge. Es sind jetzt die Worte (Programme) des Zielrechners verfügbar. Man kann sie (wie in einem gewöhnlichen Forth) durch das Nennen ihres Namens zur Ausführung bringen. Damit entsteht der Anschein, daß man nur noch auf dem Zielrechner arbeitet.

Programmierung
Aufwand auf dem Zielrechner

comFORTH-plus wurde so angelegt, daß der zusätzliche Aufwand für die Kopplung auf dem Zielrechner so gering wie möglich gehalten werden kann. Dieser besteht in der Programmierung von Routinen für:
- Schnittstelle initialisieren
- 16 Bit senden
- 16 Bit empfangen.

Dabei kann fast immer auf vorhandene Schnittstellenprozeduren zurückgegriffen werden. Weiterhin muß das Rahmenprogramm für die Kommunikation mit dem Entwicklungsrechner aufgenommen werden. Es ist von der jeweiligen Umgebung unabhängig, in Forth formuliert und belegt nur zirka 30 Byte des oft sehr eng bemessenen Speichers.

Die Software des Entwicklungsrechners

Auf dem Entwicklungsrechner befindet sich das „Hirn" des verteilten Systems. Hier sind neben den eigenen auch die Namen der Worte des Zielrechners gespeichert. Sie befinden sich ebenfalls im Wörterbuch /3/. Dadurch jedoch, daß sie über eine separate Suchordnung verwaltet werden, können sie von den Wörtern des Entwicklungsrechners unterschieden werden. Weiterhin befinden sich auf dem Entwicklungsrechner Werkzeuge zum Testen von Programmen des Zielrechners, ein Zielcompiler, mit dessen Hilfe weitere Programme in Hochsprache erzeugt werden können, sowie ein Zielassembler für den Prozessor des Zielrechners für das Erzeugen von Maschinenroutinen. Damit eignet sich comFORTH-plus nicht nur für das Testen und die Inbetriebnahme von Programmen, sondern auch zu deren Ergänzung um weitere Wörter (Programme). Das bringt eine erhebliche Verkürzung des Softwareentwicklungszyklus (Editieren, Übersetzen, Testen) mit sich.

Ausführen von Zielprogrammen
Befindet man sich im Target-Modus, kann man als Bediener auf dem Entwicklungsrechner die Wörter des Zielrechners in der forthtypischen Form durch Nennen ihres Namens ausführen. Intern befindet sich dafür der Forth-Interpreter in einem speziellen Zustand, in dem für jedes in der Suchordnung des Zielrechners gefundene Wort seine dortige Adresse ermittelt und an **TEXEC** (Bild 2) übergeben wird. Dieses prüft zuerst, ob vom Zielrechner eine Quittung als Signal der Empfangsbereitschaft vorliegt. Falls dies nicht der Fall ist, wird aktiv auf ihr Eintreffen gewartet. Damit ist die Synchronisation der beiden Rechner auf einfachste Art erreicht. Danach wird die Adresse des auszuführenden Wortes an den Zielrechner übermittelt. Er interpretiert laut seinem Rahmenprogramm **QUIT** (Bild 3) alle so empfangenen Daten als eine Codefeldadresse /1/ und ruft das sich dort befindliche Wort auf. Damit ist der gewünschte Programmstart auf dem Zielrechner durchgeführt worden. Nach dessen Bearbeitung erfolgt wieder eine Rückkehr in das Rahmenprogramm. Dieses sendet eine Quittung an den Entwicklungsrechner und geht danach erneut auf Empfang.
In der Zeit zwischen dem Aussenden der Adresse und dem Empfang der Quittung kann sich der Entwicklungsrechner auch anderen Aufgaben widmen. Damit ist eine parallele Programmierarbeit auf beiden Rechnern möglich. Im Rahmen des Testens und der Inbetriebnahme von Software ist dieser Fakt allerdings von nebensächlicher Bedeutung und soll deshalb hier nicht weiter betrachtet werden.

Übermitteln von Daten
Wenn in Forth im interpretativen Zustand Zahlen eingegeben werden, so gelangen sie als 16-Bit-Daten auf den Parameterstapel. Dort werden sie von den Forth-Worte (Programmen) erwartet oder als Ergebnis hinterlassen. In funktioneller Analogie werden im Target-Modus die Zahlen auf den Stapel des Zielrechners gelegt.
Die Möglichkeit der Übermittlung von Daten zwischen Eintwicklungs- und Zielrechner muß demnach ebenfalls vorhanden sein. Forth kommt uns hier mit seiner Strategie der Problemlösung schon auf halbem Wege entgegen. Eine Datenübertragung ist nämlich nichts anderes, als 16 Bit auf der einen Seite (vom Stapel) zu senden und auf der anderen (zum Stapel) zu empfangen. So etwas läßt sich auf der Basis von TEXEC programmieren (siehe Bild 4).
Für das Senden von Daten zum Zielrechner (PUT) wird auf diesem das Wort *16 Bit empfangen* aktiviert. Auf dem Entwicklungsrechner brauchen die nächsten 16 Bit nur zusätzlich gesendet zu werden. Für die Übertragung in der entgegengesetzten Richtung (GET) wird auf dem Zielrechner das Wort *16 Bit senden* gestartet und auf dem Entwicklungsrechner der Empfang veranlaßt. Dieses Verfahren verlangt auf dem Zielrechner kein zusätzliches Programm. Der Nachteil besteht in der uneffektiven Benutzung der Schnittstelle, da zu jedem Datentransfer zwei zusätzliche Übertragungen (Adresse des Wortes und die Quittung) hinzukommen.

Erzeugen von Programmen
Zum Erzeugen von Programmen auf dem Zeilrechner stehen ein Zielcompiler zur Programmierung in Forth und ein Zielassembler für die Maschinenprogrammierung zur Verfügung. Beide laufen, wie schon erwähnt, auf dem Entwicklungsrechner. Da der Zielassembler nur Maschinencode für den Zielprozessor zu erzeugen hat, ist er von der CPU des Entwicklungsrechners unabhängig.
Bevor mit dem Erzeugen von Zielprogrammen begonnen werden kann, muß ermittelt werden, wo der neue Code auf dem Zielrech-

```
   Datei C:WIRT.CF2      Block 1
 0 \ TEXEC                                                    RN 03-Jan-88
 1
 2 : TEXEC   ( ps: 'cf ===> ; veranlasst Ausfuehrung des Wortes   )
 3           (                 mit 'cf auf dem Zielrechner         )
 4   AUX>              \ Empfang der Quittung [evtl. warten]
 5   TEXEC?            \ Quittung auswerten [evtl. Fehlernachricht]
 6   >AUX              \ 'cf zum Zielrechner abschicken
 7   ;                 \ Definition beendet
 8
 9
10
11
12
13
14
15
```

Bild 2 Interpreteranschluß für das Starten von Zielprogrammen

```
   Datei C:ZIEL.CF2      Block 1
 0 \ QUIT                                                     RN 03-Jan-89
 1
 2 : QUIT  ( ps: ===> ; Rahmenprogramm fuer den Zielrechner )
 3   RP0 RP!          \ gehoert zur FORTH-typischen Initialisierung
 4   0 >AUX           \ Dummy-Quittung abschicken
 5   BEGIN            \ Anfang der Endlosschleife
 6      AUX>          \ 16 Bit empfangen [evtl. warten]
 7      EXECUTE       \ als Adresse ansehen & ausfuehren
 8      0 >AUX        \ Quittung als Fertigmeldung zurueckschicken
 9   AGAIN            \ unbedingter Ruecksprung zu BEGIN
10   ;
11
12
13
14
15
```

Bild 3 Rahmenprogramm des Zielrechners

ner plaziert werden soll. Alle weiteren notwendigen Informationen, zum Beispiel die Adressen anderer Worte (Programmodule), befinden sich bereits auf dem Entwicklungsrechner.
Während des Übersetzungsvorganges wird der Programmcode automatisch zum Zielrechner übertragen und dort in den Speicher eingeschrieben. Nach syntaktisch korrektem Abschluß erhalten die neuen Worte einen gültigen Eintrag im Wörterbuch des Entwicklungsrechners und sind somit aufrufbar.
Weiterhin besteht die Möglichkeit, Programme für den Entwicklungsrechner zu erzeugen, die sich auf Worte des Entwicklungs- und des Zielrechners beziehen. Am Beispiel einer schnellen Blockübertragung soll das näher erläutert werden (siehe Bild 5).
Das Wort **BPUT** (Block 1) wird für den Zielrechner definiert, daß heißt, daß der Programmcode dorthin übertragen und im Zielrechner abgelegt wird. Der Wörterbucheintrag zu BPUT bleibt auf dem Entwicklungsrechner. Wird BPUT vom Entwicklungsrechner aus aufgerufen, so arbeitet es auf dem Zielrechner.
In Block 2 wurde **BGET** definiert. Dieses Wort verbleibt komplett im Entwicklungsrechner und arbeitet bei seinem Aufruf auch hier. In BGET veranlaßt das Wort **TARGET**, daß ab jetzt Wörter des Zielrechners (hier BPUT) benutzt werden sollen. Mit **HOST** wird wieder auf die Verwendung von Wörtern des Entwicklungsrechners zurückgeschaltet. Mit dem Aufruf von BGET erreicht man, daß zuerst die für BPUT notwendigen Parameter übertragen werden und danach BPUT auf dem Zielrechner zur Abarbeitung gebracht wird. Die folgenden Instruktionen laufen dann ausschließlich auf dem Entwicklungsrechner.

Entwicklungswerkzeug Cross-Compiler

Damit der Zielrechner überhaupt mit dem Entwicklungsrechner kommunizieren kann, benötigt er mindestens das Rahmenprogramm **QUIT** (Bild 3) und die darin verwendeten Wörter. Für dessen Erzeugung ist ein weiteres Softwarewerkzeug erforderlich. Im Rahmen der Gesamtkonzeption kommt dafür nur ein Forth-Compiler in die engere Auswahl. Der *comFORTH-CrossCompiler* /5/ wird dieser Aufgabe gerecht. Er kann neben interaktiven Forth-Basissystemen (comFORTH wurde ebenfalls mit Hilfe dieses Cross-Compilers generiert) auch kleine Forth-Programme erzeugen. Eine funktionelle Aufteilung des Codes in verschiedene Speichersegmente, unter anderem auch eine ROM-RAM-Trennung, ist möglich. Für die Arbeit mit comFORTH-plus wird die Möglichkeit des Separierens der Wortköpfe (Wortname) in ein extra Speichersegment (bedeutet für den Cross-Compiler das Erzeugen einer separaten Datei) ausgenutzt. Sie sind ja der Teil des Zielprogramms, der auf dem Entwicklungsrechner benötigt wird.
Wurde die Applikation für den Zielrechner fertig ausgetestet und als praxistauglich befunden, wird sie noch einmal komplett mit Hilfe des Cross-Compilers als abgeschlossene Lösung übersetzt und ist damit einsatzfähig.

Ausblick

Der Beitrag zeigt, wie durch eine Rechnerkopplung mit einer einfachen Schnittstellenbedienung ein beeindruckender Durchgriff vom Entwicklungs- auf den Zielrechner möglich ist. Da sich fast der gesamte Softwareaufwand auf dem Entwicklungsrechner befindet, sind die Ansprüche hinsichtlich der Zielhardware minimal.
comFORTH-plus entfaltet mit seiner Umgebung eine außergewöhnliche Effektivität, die die Programmentwicklungszeiten für Zielrechner stark schrumpfen läßt. Das ist vor allem auf die enorme Verkürzung des Softwareentwicklungszyklus zurückzuführen, die dadurch eintritt, daß soeben übersetzte Worte (Programmteile) unmittelbar gestartet, korrigiert und wieder neu übersetzt werden können. Als Beispiel soll die Softwareentwicklung für eine an der WPU entwickelte intelligente Grafikkarte mit Z 80-CPU dienen, die vorher mit Hilfe eines Macroassemblers programmiert wurde. Nach Einführung des Werkzeuges comFORTH-plus erhöhte sich die Effektivität der Programmierung etwa auf das 5fache. Dadurch konnte die Entwicklung einer GSX-Firmware vorgenommen werden. comFORTH-plus ist ausschließlich in Forth geschrieben und wurde auf das Basissystem comFORTH 2 nachgeladen /6/. Da es sich nur an den forthinternen Schnittstellen orientiert, ist es sofort überall dort einsetzbar, wo comFORTH 2 /7/ läuft. Das sind alle Rechner mit den Betriebssystemen CP/M-80 (Versionen 2.2 und 3.0), CP/M-86 (Version 2.2) und MS-DOS (ab Version 3.0).

Literatur

/1/ Brodie, L.: Programmieren in FORTH. München: Carl Hanser Verlag 1986
/2/ Woitzel, E.: comFORTH – Programmierwerkzeug FORTH unter SCP. edv-aspekte, Berlin 5 (1986) 4, S. 47
/3/ Höhenleitner, T.: FORTH – eine moderne Softwarephilosophie. Mikroprozessortechnik, Berlin 2 (1988) 2, S. 35
/4/ Vack, G.-U.: FORTH: Eine außergewöhnliche Softwarekonzeption. Mikroprozessortechnik, Berlin 1 (1987) 6, S. 163
/5/ Neuthe, R.: CrossCompiler für FORTH. Vortrag auf der Fachtagung FORTH, Suhl 1988
/6/ Woitzel, E.; Neuthe, R.: Programmierung zugeschnittener Rechnerkonfigurationen unter FORTH. Postervortrag auf dem 33. Internationalen Wissenschaftlichen Kolloquium der Technischen Hochschule Ilmenau 1988
/7/ Woitzel, E.: Neue Leistungsmerkmale des Basissystems comFORTH 2.0. Vortrag auf der Fachtagung FORHT, Suhl 1988

✉ KONTAKT ☎

Wilhelm-Pieck-Universität Rostock, Sektion Technische Elektronik, Wissenschaftsbereich Automatische Steuerungen, Albert-Einstein-Straße 2, Rostock 6, 2500; Tel. 40 52 02


```
: PUT ( ps: 16b ===> ; Transfer zum Zielrechner )
     TARGET AUX> HOST >AUX ;

: GET ( ps: ===> 16b ; Transfer vom Zielrechner )
     TARGET >AUX HOST AUX> ;
```

***Bild 4 Programmvariante für* PUT *und* GET**

```
  Datei C:TRANSFER.CF2 Block 1
 0 \ Transfer vom HOST                              RN 22-Dez-88
 1 \ Uebertragung von n Byte ab Adresse "Anfang"
 2
 3  TARGET               \ Beginn der Arbeit auf dem Zielrechner
 4 : BPUT  ( ps: Anfang n ===> ; Block senden )
 5   OVER                \ Kopieren von "Anfang" zur Stapelspitze
 6   +                   \ Addition "Anfang + n"
 7   SWAP                \ Vertauschung von "Anfang" und "Anfang+n"
 8   DO                  \ Beginn der Schleife, Durchlauf von
 9                       \ "Anfang" bis "Anfang+n-1"
10      I                \ Schleifenindex = Adresse
11      C@               \ Byte von Adresse lesen
12      >AUX             \ und zur Schnittstelle abschicken
13   LOOP                \ Ende der Schleife
14 ;                     \ Ende der Definition
15

  Datei C:TRANSFER.CF2 Block 2
 0 \ Empfang vom TARGET                             RN 22-Dez-88
 1 \ Empfang von n Byte ab Adresse "TAnfang" und Ablage ab Adresse
 2 \ "HAnfang" im Wirtsrechner
 3
 4  HOST                 \ Arbeit auf dem Wirtsrechner
 5 : BGET  ( ps: HAnfang n TAnfang ===> ; Block empfangen )
 6   PUT                 \ "TAnfang" zum Zielrechner uebermitteln
 7   DUP PUT             \ "n" duplizieren und abschicken
 8   TARGET BPUT HOST \ "BPUT" auf dem Zielrechner anstossen und
 9                       \ folgende Aktionen wieder auf dem Wirt
10   OVER + SWAP         \ Parameter fuer die Schleife bereitstellen
11   DO                  \ Beginn der Schleife
12      AUX>             \ Byte empfangen
13      I                \ Schleifenindex = Adresse
14      C!               \ Byte auf Adresse abspeichern
15   LOOP ;
```

Bild 5 Programm für eine schnelle Blockübertragung

TERMINE

1. Wissenschaftliche Konferenz INFORMATIK

WER? Informatik-Zentrum des Hochschulwesens an der Technischen Universität Dresden
WANN? 22. bis 23. Februar 1990
WO? Dresden
WAS?
- Softwaretechnologie und Programmiersprachen
- Rechnernetze
- Informatik und CIM
- Künstliche Intelligenz

WIE? Anfragen richten Sie bitte an den Veranstalter: Technische Universität Dresden, Informatik-Zentrum, Konferenz INFORMATIK, Tagungsbüro, Mommsenstraße 13, Dresden, 8027; Tel. 4 57 53 38.

Prof. Dr. Tzschoppe

Wissenschaftliche Tage 1990

WER? Technische Universität „Otto von Guericke" Magdeburg
WANN? 3. bis 13. September 1990
WO? Magdeburg
WAS?
- VI. Symposium „Zuverlässigkeit" vom 3. bis 5. September
- 4. Fachtagung „Gestaltung von Fertigungsprozessen im Maschinenbau" vom 5. bis 6. September
- 8. Vortragstagung „Fertigung und Gütesicherung im Zahnradgetriebebau" vom 12. bis 13. September

WIE? Teilnahmemeldungen richten Sie bitte an: Technische Universität „Otto von Guericke" Magdeburg, Tagungsbüro, PSF 124, Magdeburg, 3010

Prof. Dr. Probst

Computer-Club

MS-DOS-Tip

Tastaturbelegung ändern

Wenn in die Datei CONFIG.SYS die Befehlszeile DEVICE = ANSI.SYS integriert wird, stehen erweiterte Möglichkeiten zur Bildschirmsteuerung zur Verfügung. Dabei kommen dann ANSI-Steuerfolgen zur Anwendung. Diese sind bindend, falls sie durch DOS-Aufrufe ausgesendet werden, welche die Standardeingabe- bzw. -ausgabeeinheit oder die Standardfehlerausgabeeinheit benutzen. Dies sind die Funktionsaufrufe 01H, 04H, 06H, 07H, 09H und 40H. Die Steuerfolgen für die ANSI-Treiber sind keine Befehle oder Kommandos für MS-DOS und können nicht in der CONFIG.SYS- oder der AUTOEXEC.BAT-Datei als Kommandozeilen stehen.

Für die Umkodierung der Tastaturbelegungen sind im ANSI-Standard folgende Steuerfolgen enthalten:

ESC [#; #; . . ; #P
ESC [„KETTE" P
ESC [#; „KETTE"; #;
#; „KETTE"; #P

oder jede beliebige Kombination von Zeichenkette und Dezimalwert eines Zeichens.

Der erste ASCII-Kode in der Steuerfolge definiert die neu zu belegende Taste. Die restlichen Zeichen definieren die Zeichenfolge, die von der Taste neu geliefert werden soll. (Beachte: Ist der erste Kode in der Steuerfolge 0 (00H), dann bilden der erste und der zweite Kode zusammen den *erweiterten* ASCII-Kode).

Die Ausführung der ESC-Steuerfolgen kann erreicht werden durch:

- eine Datei, die die Steuerfolgen enthält und die Darstellung dieser auf dem Bildschirm mit dem Kommando TYPE oder COPY;
- Ausführung der Kommandos nach der MS-DOS-Bereitschaftsanzeige.

Beispiele:

1. Umdefinieren der Taste A bzw. a zu Q bzw. q:

Erzeugen einer Datei:
ESC [65; 81P → A wird Q
ESC [97; 113P → a wird q

Nach MS-DOS-Bereitschaftsanzeige:
prompt $e [65; 81P → A wird Q
prompt $e [97; 113P → a wird q

2. Belegen der Taste F10 mit der Zeichenkette *dir* und einem nachfolgendem Wagenrücklauf:

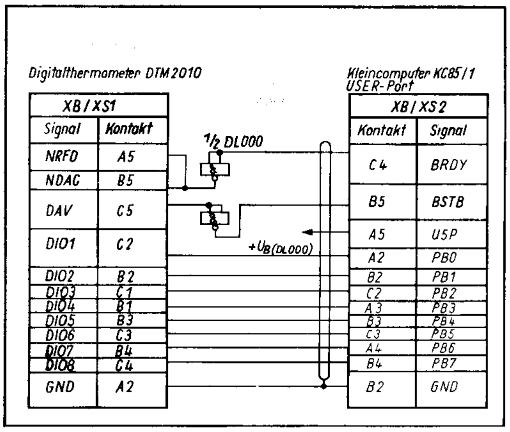
Erzeugen einer Datei:
ESC [0; 68; „dir"; 13 p

Nach MS-DOS-Bereitschaftsanzeige:
promt $e [0; 68; „dir"; 13 p

das Metasymbol $e ist die Darstellung von ESC (1BH) im PROMPT-Kommando. Der erweiterte ASCII-Kode für die Taste F10 ist 0 und 68. Für den Wagenrücklauf steht 13 (ODH).

Dimiter Mantschew

Temperaturmessung mit dem KC 85/1



Für Laborversuche bestand die Aufgabe, das Digitalthermometer DTM 2010 mit dem Kleincomputer KC 85/1 zu koppeln.

Die Temperaturmeßwerte sollen vom Kleincomputer abgefragt, gespeichert und verarbeitet werden. Zur Meßwertabfrage soll ein anwenderfreundliches Basic-Programm eingesetzt werden, das den numerischen Temperaturmeßwert einer Variablen zuordnet.

Die Ausgabe des gemessenen Temperaturwertes erfolgt entsprechend der Anwenderdokumentation des Digitalthermometers DTM 2010 /1/ mit einem dem IMS-2-Standard ähnlichen Verfahren. Das Meßergebnis wird byte-seriell, bit-parallel ausgegeben. Dabei entspricht der Ausgang DIO 1 dem Bit (2^7) und DIO 8 dem Bit (2^0) des Datenbytes. Zur Steuerung sind drei Leitungen mit den Signalbezeichnungen NDAC, NRFD (Eingänge) und DAV (Ausgang) vorhanden.

Für diese Signale wurden nachfolgende Funktionen ermittelt:

NDAC = negierte Datenempfangsquittung, aktiv mit H-L-Flanke
NRFD = negierte Datenanforderung, aktiv mit L-H-Flanke
DAV = Sendedaten gültig, aktiv mit H-L-Flanke

Alle Signale sind mit dem Lastfaktor 1 TTL-kompatibel.

Der Anschluß des Digitalthermometers DTM 2010 an den Kleincomputer KC 85/1 erfolgt über die Hardware nach Bild 1, die keine Eingriffe in die Geräte erfordert. Die Betriebsspannung von + 5 V erhält der Schaltkreis DL 000 vom Kleincomputer.

Er ist in der Griffschale des Steckers XS 2 in Freiverdrahtung eingebaut.

Das Verbindungskabel zwischen Thermometer und Kleincomputer ist im vorliegenden Fall 55 cm lang. Damit Störungen unterdrückt werden, sind alle freien Adern und der Kabelschirm einseitig im Stecker XS 2 auf Masse geschaltet.

Als Eingang am Kleincomputer KC 85/1 wird der PIO-Port des Useranschlusses /2/ genutzt.

Das Basic-Programm ist als Unterprogramm für die häufig genutzte Betriebsart Mittelwertmessung des Digitalthermometers DTM 2010 geschrieben. Leider ist die Synchronisation des Programmes mit dem Basic-Befehl WAIT nicht möglich.

Als Alternative wird nach der Teilzeichenkette „ MIT" für den Beginn der gültigen (störungsfreien) Meßwertübernahme gesucht. Wegen des asynchronen Starts des Programms sind mehrere Abfragen notwendig, bis die Teilzeichenkette „ MIT" gefunden wird. Die Folge sind unregelmäßige Programmlaufzeiten im Bereich 0,5 Sekunden bis 3 Sekunden für das Einlesen des Temperaturmeßwertes.

Das vorgestellte Beispielprogramm, welches die Abfrage des Digitalthermometers DTM 2010 im Unterprogramm durchführt, ereichte durchschnittlich 20 Messungen je Minute.

Ist die Meßwertübernahme aus anderen Betriebsarten des Digitalthermometers DTM 2010 erforderlich, so ist in der Zeile 70 des Basic-Programmes die entsprechende Teilzeichenkette (z. B. „ MAX" für Maximalwert) einzutragen.

Die Zeichenkette fünf Leerzeichen (Normalbetrieb) ist nicht auswertbar, weil die Vornullen des Meßwertes in Leerzeichen umgewandelt werden.

```
10 RUN 100
20 OUT 139,7
30 X$ = ""
40 FOR I = 1 TO 16
50 X$ = X$ + CHR$ (255-INP (137))
60 NEXT
70 IF LEFT $ (X$, 5) = „     MIT"
   THEN Y = VAL (RIGTH $
   (X$, 10)) : RETURN : ELSE 30
100 CLS : PRINT „Aktuelle Tem-
   peratur:"
110 DIM X$ 16
120 GOSUB 20 : PRINT AT (5,7);
   „C      " : GOTO 120
```

Literatur

/1/ VEB Thermometerwerk Geraberg: Anwenderdokumentation Digitalthermometer DTM 2010, DTM 2020, DTM 2030. Geraberg 1985

/2/ VEB Robotron-Meßelektronik „Otto Schön" Dresden: Bedienungsanleitung Kleincomputer Z 9001 Dresden 1985

Bernd Voitel

KONTAKT

Ingenieurschule für Elektrotechnik und Maschinenbau Eisleben, Geiststraße 2, Eisleben, 4250; Tel. 54 84 App. 77

Nutzung der Fenstervektoren für Basic

Das Betriebssystem CAOS des KC 85/3 ermöglicht es, im Fenstervektorspeicher B99CH . . . B9FFH mit jeweils zehn Byte die Parameter von zehn verschiedenen Bildschirmfenstern abzulegen. Diese können jederzeit mit ihrer Nummer (0 . . . 9) aufgerufen werden, wobei der zuvor gültige Fenstervektor zurückgespeichert wird.

Wenn ein Basic-Bildschirmdesign mit mehreren Fenstern gestaltet werden soll, zwischen denen mehrfach gewechselt wird und die auch noch unterschiedliche Farben haben mögen, müssen jedesmal die Anweisungen WINDOW und COLOR (mit Parametern) programmiert werden. Das ist umständlich und mit dem Einsatz der Fenstervektoren eleganter lösbar.

Im Listing sind die zu programmierenden Einzelheiten zusammengestellt. Am Anfang des Basic-Programms werden die Parameter aller später benötigten Fenster mittels VPOKE in den Fenstervektorspeicher eingetragen. Die Bedeutung der Parameter, die in DATA-Zeilen einzusetzen sind, ist /1/ zu entnehmen. Die Fensternummer 0 muß dem vom Betriebssystem initialisierten Fenster vorbehalten bleiben! Danach wird die Adresse eines zum Aufrufen benötigten MC-Programms in eine von der USR-Funk-

Tafel 1 Aufbau des Fenstervektors

Byte	Bedeutung
[illegible]	Fensteranfang (Spalte)
[illegible]	Fensteranfang (Zeile)
[illegible]	Fenstergröße (Spalte)
[illegible]	Fenstergröße (Zeile)
[illegible]	Kursor (Spalte)
[illegible]	Kursor (Zeile)
[illegible]	Steuerbyte
[illegible]	Farbe
[illegible]	Reaktion auf das Fensterende (aus 0B7A4/5)

BASIC-Programm

```
10 ! EINTRAGUNG IN DEN FENSTERVEKTORSPEICHER
15 ! FH = Höchste Fensternummer (1...9)
20 FOR I=1 TO FH : FOR J=0 TO 9
30 READ B : VPOKE 14748+10*I+J,B
40 NEXT J : NEXT I : VPOKE 14235,0
50 DATA ...          : ! Parameter f. Fenster 1
60 DATA ...          : ! Parameter f. Fenster 2
:
100 ! MC-ADRESSE EINTRAGEN
110 DOKE 772,MC
:
200 ! AUFRUFEN EINES FENSTERS
205 ! FX = Fensternummer
210 D=USR(FX) : ! D = Dummyvariable
```

Maschinencode-Programm

```
CD6FC9  MC  CALL 0C96FH   ;Parameterübergabe
7B          LD   A,E      ;Fensternr. nach A
1E3D        LD   E,3DH    ;CAOS-UP Nr.3DH
CD0CF0      CALL 0F015H   ;Programmvert. 5
C9          RET
```

tion vorgegebene Speicherzelle geschrieben. Nun kann an beliebiger Stelle im Programm über D=USR(FX) der Aufruf eines der initialisierten Fenster erfolgen. Die Variable D muß hier nur wegen des vorgeschriebenen Syntax eingeführt werden, ihr Wert ist uninteressant. Das kleine MC-Programm wird zweckmäßigerweise in einer REM-Zeile untergebracht. Es vermittelt die Parameterübergabe vom Basic- zum MC-Programm und ruft das CAOS-UP 3DH, mit dem schließlich der Aufruf des Fensters erfolgt.
Bemerkenswert ist noch folgender Unterschied: Beim Aufrufen des Fensters mit WINDOW springt der Cursor stets in die HOME-Position. Der Fenstervektor dagegen speichert die aktuelle Cursorposition, so daß beim Wiederaufruf eine begonnene Ausschrift fortgesetzt werden kann!

Wolfram Schütze

Gelöschte Basic-Programme erreichen

Durch NEW gelöschte Basic-Programme werden durch das Programm HELP (siehe Hexdump) für den KC 85/3 wieder erreichbar. Es wird aus dem Menü bzw. in Basic mit CALL* <Ladeadresse> gestartet. Bei der Verwendung eines KC 85/2 mit aktiviertem Basic-Modul kann das Programm nur aus dem Menü unter Angabe der Anfangsadresse des Basic-Textes (2C00H) gestartet werden.
Während der Abarbeitung wird auf ein vorhandenes Basic-Programm getestet, anschließend werden die entsprechenden Zeiger gesetzt.

Thomas Fleischer

```
%DISPLAY  0000 0067
0000   18 0D 7F 7F 48 45 4C 50
0008   01 3A 81 B7 2A 82 B7 F5
0010   DB 88 CB 97 D3 88 F1 FE
0018   01 28 03 21 00 04 E5 FD
0020   E1 11 05 00 19 7E FE 7F
0028   38 35 AF 01 FF BA ED B1
0030   20 2D FD 75 01 FD 74 02
0038   ED B1 20 23 16 02 0B BE
0040   20 F6 23 15 20 F8 FD 75
0048   D7 FD 74 D8 FD 75 D9 FD
0050   74 DA FD 75 DB FD 74 DC
0058   DB 88 CB D7 D3 88 C9 CD
0060   58 00 CD 03 F0 19 C9 00
%
```

Umleitung für SUBMIT noch schneller

Für die zahlreichen Zuschriften und Hinweise zu dem Beitrag von Frank Isekeit in MP 7/89, Seite 205, bedanken wir uns bei allen Lesern. Die darin enthaltenen Lösungen lassen sich im wesentlichen so zusammenfassen: Es ist nicht erforderlich, das Programm SUBMIT zu ändern; mit den folgenden Kommandos wird auch die Umschaltung auf die RAM-Disk erreicht:
PIP E: = A : PIP.CMD
E : PIP E : = A : $$$$$. SUB
E:
ERA A : $$$$$. SUB
...
Besonderer Dank gilt Herrn Kämpfe aus Markkleeberg, Herrn Wölffer aus Erfurt, Herrn Schmidt aus Bitterfeld und Herrn Böhm aus Karl-Marx-Stadt.

Redaktion MP

Textverarbeitung von Datenbanken

Dr. Sylvia Springer, Knut Glockmann

Viele Anwender von dBase II wollen Daten aus dem Datenbanksystem im Textverarbeitungssystem WORDSTAR weiter verarbeiten. In den entsprechenden Programmdokumentationen ist hierzu wenig erklärt oder es fällt schwer, die Erklärungen richtig anzuwenden.
Deshalb sei ein kleines Beispiel zur Demonstration angeführt. Die Datenbank nach Bild 1 enthält einige Probedatensätze einer Personaldatei. Alle Datenfelder sind vom Typ C, die Länge der Datenfelder ist gleichgültig.
Die Aufgabe könnte darin bestehen, an die Mitglieder der Arbeitsgruppe CAD/CAM Einladungsschreiben zu regelmäßig stattfindenden Arbeitsgruppenberatungen zu senden. Für diese Schreiben werden nur die Datenfelder *Anrede Name Vorname Gruppe* benötigt, da die Einladungen betriebsintern verschickt werden. Zuerst müssen die benötigten Datenfelder der zutreffenden Datensätze in eine Textdatei (EINDAT.TXT) kopiert werden, die dann beim Kombodruck gelesen wird. Hierzu benutzen wir den im Bild 2 gezeigten dBase-Befehl. Die beim Kopieren in eine Textdatei normalerweise anzugebende SDF-Klausel ist bereits in dem Befehl *delimited* enthalten.
Danach wird dBase verlassen. Durch *type EINDAT.TXT* kann man sich vom ordnungsgemäßen Aufbau der Einladungsdatei EINDAT.TXT überzeugen. Die einzelnen Datenfelder sind durch Kommas getrennt, jeder Datensatz wird mit einer Zeilenschaltung abgeschlossen.
Im Beispiel haben wir neben der Anwendung der Bedingungsklausel (for gruppe = 'CAD/CAM') auch die Bereichsklausel (field name, vorname, ..) verwendet. Fehlen diese beiden Klauseln, so werden alle Felder aller Datensätze in eine Textdatei kopiert.
Die angelegte Textdatei wird nun als Datenfile für den Kombo-Druck verwendet. Im Bild 3 ist ein Beispiel eines solchen Kombo-Druck-Textes dargestellt
Um den Grundtext des Schriftstückes für alle Arbeitsgruppenberatungen verwenden zu können, sollen die Variablen *DATUM*, *ZEIT* und *ORT* im Dialog eingegeben werden. Diese Angaben sollen jedoch nicht bei jedem Exemplar eingegeben werden. Deshalb arbeiten wir mit einer Rahmendatei (RAHMEN.TXT), in die wir beim Kombo-Druck die Datei *BEISPIEL.TXT* einfügen lassen.
Bild 4 zeigt den Aufbau der Rahmendatei.
Damit sind alle Vorbereitungen getroffen. Die Datei RAHMEN.TXT wird nun mit dem Kommando **M** gedruckt. Nach Anzeige der Kommentarzeile *Einladung zur Arbeitsgruppenberatung* werden die über die **.av**-Kommandos vereinbarten Variablen einmal im Dialog eingegeben. Mit dem Kommando **.fi** wird die Datei BEISPIEL.TXT eingefügt, die wiederum aus der Datei EINDAT.TXT die Variablen für die Personen holt. Es werden so viele Schriftstücke gedruckt, wie Datensätze vorhanden sind. Es muß aber auf die exakte Zuordnung der Anzahl und der Reihenfolge der Datenfelder und der zu lesenden Variablen geachtet werden, da sonst die Anordnung der Variablen im Text durcheinander kommt. Soll jedes Schriftstück auf ein neues Blatt gedruckt werden, schließt man die Datei mit dem Kommando **.pa** ab.
Die letzte Zeile der Datei BEISPIEL.TXT muß auf jeden Fall mit einem harten Zeilenende (CR) abgeschlossen werden. Wird dies vergessen, beginnt der Drucker das zweite und die folgenden Exemplare fehlerhaft (meist mit Punkt-Kommandos).
In unserem Beispiel müssen sich alle Dateien auf dem Vorranglaufwerk befinden. Wollen Sie mit mehreren Laufwerken arbeiten, so muß die Laufwerksbezeichnung vor den Dateinamen bei den Kommandos **.fi** und **.df** angegeben werden (z. B. .fi B:EINDAT.TXT). Wir hoffen, daß wir Ihnen mit diesem kleinen Beispiel helfen, Ihre Datenbanken auch für die Textverarbeitung nutzen zu können.

Anrede	Name	Vorname	Strasse	Ort	PLZ	Gruppe
Koll.	Werner	Peter	Lindenstrasse 1	Rosendorf	1234	CAD/CAM
Gen.	Meier	Gerd	Kastanienallee 2	Nelkenstadt	2345	Standardisierung
Kolln.	Abel	Rita	Eichenweg 3	Tulpenstadt	3456	CAD/CAM
Kolln.	Bartsch	Rosi	Buchenplatz 4	Liliendorf	4567	Neuererwesen

Bild 1 Beispiel einer dBase II-Datenbank

```
copy to EINDAT.TXT field anrede,name,;
vorname,gruppe for gruppe='CAD/CAM'   ;
delimited with ,
```

Bild 2 dBase-Kommando zum Kopieren einer Datenbank in eine Textdatei

```
.df EINDAT.TXT
.rv anrede,name,vorname,gruppe
             Werte(r) &anrede& &name&, &vorname&!

Am &datum& findet um &zeit& Uhr im &ort& unsere nächste Be
ratung  der  Arbeitsgruppe &gruppe& statt.  Wir  laden  Sie
hiermit ein und erwarten Ihre Teilnahme.

                                     Mit soz. Gruß
                                      gez. Meier
                                    Betriebsdirektor
```

Bild 3 Beispieldatei BEISPIEL.TXT für den Kombo-Druck

```
.cs Einladung zur Arbeitsgruppenberatung
.av 'Bitte Datum eingeben : ',datum
.av 'Bitte Uhrzeit eingeben : ',zeit
.av 'Bitte Ort eingeben : ',ort
.fi BEISPIEL.TXT
```

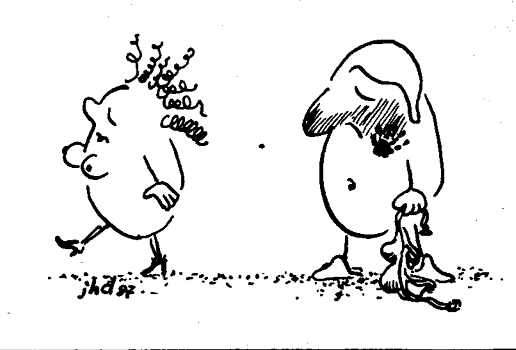
Bild 4 Rahmendatei RAHMEN.TXT zum Druck von Einladungen

KONTAKT

Technische Universität „Otto von Guericke", CAD/CAM-Zentrum, Boleslaw-Bierut-Platz 5, Magdeburg, 3032; Tel. Magdeburg 59 29 97, Dr. Springer

Kleines Lexikon der Mikrorechentechnik

Q
wie Quittung



R
wie Rotationsbefehl

Basissystem comFORTH 2

Mit dem Basissystem comFORTH 2 wird dem Anwender ein professionelles Forth-Enwicklungssystem zur Verfügung gestellt, das auf den meisten der im Inland eingesetzten PC-Systeme kompatibel lauffähig ist (A 5120, PC 1715, A 71xx, EC 1834, PC-XT/AT-kompatible).
comFORTH 2 erfüllt die Kriterien des Standards Forth-83 und berücksichtigt die durch das ANSI vorgelegten Erweiterungen. Der Wortschatz des Kernsystems umfaßt etwa 1 000 Worte, die den Standard vor allem um Funktionen zur Dateiarbeit, Gerätesteuerung und Zeichenkettenverarbeitung erweitern. Die Übersetzungsgeschwindigkeit wurde gegenüber dem Vorläufersystem auf ein Mehrfaches gesteigert. Die Wörterbuchverwaltung gestattet die temporäre Aufnahme von Compilererweiterungen und die Verwendung eines Modulkonstrukts.
Zum Lieferumfang gehören neben dem Kernsystem ein Wordstar-kompatibler Editor, ein Assembler, ein Debugger sowie eine große Anzahl von Utilityprogrammen (Datenstrukturierung, Dateneingabe, bedingte Compilation u. v. m.). Bitte fordern Sie weiteres Informationsmaterial an.

Wilhelm-Pieck-Universität Rostock, Sektion Technische Elektronik, WB Automatische Steuerungen, Albert-Einstein-Straße 2, Rostock, 2500; Tel. 40 52 02

Dr. Pfüller

Diskettenarbeit am P8000

Im Rahmen unserer Forschungsarbeiten entstanden folgende zwei Programme zur Diskettenarbeit am P8000 unter dem Betriebssystem WEGA:

cpmd
Programm zum Lesen und Schreiben von CP/M-Disketten, Nutzung der Treiber /dev/fdx96ds9 und fdx96ds16 für Disketten im SCP-oder Vortex-Format. Neben der Anzeige der Directory, Kopieren von und zur Festplatte besteht die Möglichkeit, auf der Diskette Files zu löschen sowie Filegrößen und Diskettenbelegung zu ermitteln.

dosd
Mit diesem Programm können MS-DOS-Disketten gelesen und geschrieben werden. Mögliche Formate sind 180 KByte (9 × 512 × 40 SS), 360 KByte (9 × 512 × 40 DS) und 720 KByte (9 × 512 × 80 DS). Es können Files von/zur Festplatte kopiert werden, Belegungen und Filelängen ermittelt und ein Cluster editiert werden. Bei diesem Programm kann mit Pfadnamen gearbeitet werden (Wechsel der Directory).
Die Anwendung und die installierten Kommandos sind ähnlich denen, die beim CP/M-Programm POWER zur Verfügung stehen. Auf dem P8000 müssen dazu die Diskettentreiber mit 9 und 16 Sektoren/Spur zur Verfügung stehen.

Humboldt-Universität zu Berlin, Sektion Elektronik, Bereich 5, Invalidenstraße 110, Berlin, 1040

Jäkel

Kommandogenerator für DCP

Der Kommandogenerator **KG** ist eine Kombination von Benutzeroberfläche und On-line-Hilfssystem für DCP-Kommandos. Er ist für die interaktive Arbeit mit 16-Bit-PCs unter DCP (A 7150, EC 1834) gedacht und stützt die Ausführung der wichtigsten Betriebssystemkommandos. Zu jedem Kommando erhält der Nutzer Informationen über die korrekte Befehlssyntax sowie die verfügbaren Parameter und Schalter, Laufwerkswechsel und Start von Anwenderprogrammen sind unter KG ebenfalls möglich. Selbst Ungeübten wird damit die Nutzung des PCs ermöglicht, zumal in vielen Fällen Beispiele angefügt sind. Nur selten wird der Blick ins DCP-Handbuch nötig sein.
Das Programm belegt ca. 45 KByte Hauptspeicher und steht ab sofort zur Nachnutzung zur Verfügung (Nachnutzungsgebühr: 280,00 M). Interessenten senden bitte mit der formlosen Bestellung eine Diskette (formatiert) an den:

VEB Leitzentrum für Anwendungsforschung, Softwarebetrieb CAD/CAM, Betriebsteil Karl-Marx-Stadt, Palmstraße 12, Karl-Marx-Stadt, 9072; Tel. 4 63 61

Andreas

Fondsvergabeprogramm ARPASO

Das Programmpaket ARPASO kann zur quartalsweisen Vergabe beliebiger Fondssortimente nach beliebigen Vergabekriterien eingesetzt werden. Es ist einsetzbar auf jedem Rechner ab 16 Bit Verarbeitungsbreite. Der Bedarf an externer Speicherkapazität ist abhängig vom Sortimentsumfang. Die Programme des Programmpaketes belegen 500 bis 600 KByte, je nach Qualität des auf dem Rechner verfügbaren C-Compilers. Compilierungen wurden durchgeführt auf P 8000 und EC 1834. Zum Leistungsumfang des Programmpaketes gehören:
– Quartalsplanung auf der Grundlage von fei wählbaren Verteilkriterien
– Vergabe eines Artikels auf der Grundlage bereits vergebener Artikel
– Gleichzeitige Betrachtung von bis zu drei Varianten je Quartal
– Setzen von bis zu vier Davonpositionen je Quartal
– Berechnen einer zu einem bestimmten Termin verfügbaren Fondsmenge auf der Grundlage beliebiger Berechnungsformeln
– Modifizieren der verteilten Mengen durch das Setzen von Beträgen für einzelne Fondsträger
– wahlweise Davonpositionsverrechnung
– Verrechnen von Jahresendbeständen und Fortschreibungsbeständen
– Fortschreibung zu hoher Bestände
– automatische Bereinigung aller eingegebenen Fonds um die aktuell eingestellte Abpackgröße
– Reportzusammenstellung für Zwischenergebnisdateien und Endablagedateien
– Spezifikation der Artikelnummer durch eine weitere Ziffer zur Sicherung der eindeutigen Unterscheidung von Normalfonds, Zusatzfonds, Reserveauflösungen und anderen Spezialfonds
– Hintergrundverarbeitung bei allen dateiorientierten Programmen
– Weiterverarbeiten der Endablagedatei mit jeder beliebigen Datenbanksoftware
– Kommandointerpreterschnittstelle (C-Shell oder Norton) vom Hauptsteuerprogramm aus.

Ministerium für Land-, Forst- und Nahrungsgüterwirtschaft, Bereich Pflanzen- und Tierproduktion, Köpenicker Alle 39–57, Berlin, 1157; Tel. 5 05 34 51

Findeis

Digitalisiersoftware AKROK/M16

Das Grundpaket AKROK/M16 ermöglicht die rationelle maßstabsgerechte und strukturierte Digitalisierung und Änderung von Geometriedaten mittels HDG, (K 6401...K 6404) im Sinne der Primärdatenerfassung am A 7100 unter DCP. Zum Funktionsspektrum gehören:
1. Strukturiertes Erfassen der grafischen Grundelemente: Strecke, Kreis, Kreisbogen, Text, Marker und Kurve durch punktweises und kontinuierliches Digitalisieren.
2. Spezifizierung der Attribute: Farbe, Linientyp, Texthöhe und -richtung
3. beim Einlesen von Zeichnungsdaten automatische Erkennung und Konvertierung systemfremder Formate (GKS-Files, CADdy (ASCII)) in das systeminterne binäre ROK-Format
4. Ausgabe von Teilstrukturen in den Zeichnungsformaten: ROK-Format, GKS-Files, CADdy. (ASCII) und HPGL
5. Löschen und Korrektur von beliebigen Zeichnungselementen
6. Erweiterung von Zeichnungen
7. Ausgabe von Zeichnungen auf Geräte über DCP-GX.
Geplante Systemerweiterungen (mit Termin 2/90) sind:
1. Ergänzungsmodul für Transformation, Kopieren und Makro-Einfügen mit geometrischen Manipulationen wie Translation, Rotation, Skalierung und Spiegelung von Zeichnungsstrukturen
2. Datenschnittstelle IGES (AutoCAD, MEDUSA, PolyCON).
Weitere geplante Ergänzungen: Schraffur, Bemaßung, Bedienungsmodul, Einbindung weiterer grafischer Eingabegeräte und Anpassung an den EC 1834. Der Preis des Grundpaketes beträgt 7 905 M.

VEB Robotron-Elektronik Dresden, Bereich System- und Softwareapplikation, PSF 710, Karl-Marx-Stadt, 9010; Tel. 39 73 67

Schulz

Weiche für Centronics-Schnittstelle

Zur Mehrfachnutzung von Druckern an Rechnern wurde eine Schnittstellenweiche für die Centronics-Schnittstelle entwickelt. Durch den Einsatz dieser Weiche wird der Betrieb eines Druckers an zwei oder drei Rechnern möglich. Die Zuschaltung des entsprechenden Datenkanals erfolgt mit dem ersten übertragenen Datenbyte. Gleichzeitig wird der zweite Kanal für die Datenübertragung gesperrt. Von den Steuerleitungen der Centronics-Schnittstelle werden BUSY, /STROBE und /ACK verwendet. Alle anderen Steuerleitungen werden nicht durch die Weiche durchgeschaltet. Das Prinzip der Schaltung ist bei geringer Modifikation ebenso auf eine größere Anzahl von Rechnern anwendbar.
Es besteht die Möglichkeit, eine bestückte Platine für die Version mit 3 Rechnern und auch für die IFSS-Weiche (siehe MP 6/89) zu liefern.

Handelshochschule Leipzig, ORZ, Bereich Elektroniklabor, Markgrafenstraße 2, Leipzig, 7010; Tel. 74 81, App. 26/73

Hoffmann

Full-screen-Editor MEDI für Turbo-Pascal

Es handelt sich um einen Full-screen-Editor mit Wordstar-ähnlichem Leistungsumfang. Alle Bedienfolgen sind WS-kompatibel. Die Löschfunktionen und der Insertmodus sind gegenüber Wordstar etwas erweitert worden. Editiert wird ein voller Bildschirminhalt, eine Variante mit vertikalem scroll ist ebenfalls vorhanden. Die beschreibbaren Positionen sind maskiert. In der Maske können numerische und alphanumerische Felder angegeben werden, eine Erweiterung ist vorgesehen. Die Reaktion bei bestimmten Bedingungen (z. B. Zeilenende) kann über toggles eingestellt werden, ebenso wraparound und die Form des Insert-Modus.
MEDI benutzt die verfügbaren Videoattribute. Versionen verschiedenen Leistungsumfangs sind für A 7100 und PC 1715 verfügbar, eine Anpassung an andere Rechner ist leicht möglich und kann notfalls durch den Anwender vorgenommen werden. MEDI ist für Programmierer gedacht, die in ihren Programmen eine komfortable Dateneingabe benötigen. Es ist universell einsetzbar, da die Maske erst während der Laufzeit bekannt sein muß. Es wird der Quelltext (als Include-File für Turbo-Pascal 3.0) weitergegeben.

Spezialschule „Heinrich Hertz", H.-P. Lüdtke, Frankfurter Allee 14a, Berlin, 1035

Schimke

Kontomanager KM 1

Der Kontomanager KM 1 dient dem Erarbeiten, Verwalten und Auswerten des Sachkontenrahmens nach der Systematik des Staatshaushaltes der DDR (3. Ausgabe für staatliche Einrichtungen). Die komplette Systematik mit allen Sachkonten liegt als Stammdatei vor und kann für jeden Anwender spezifiziert werden. Mit dem Installationsprogramm KMINST kann der Listenkopf für den Haushaltsplan, das Paßwort, der Drucker (z. Zt. FX 80, Draft) und der Arbeitszeitfonds anwendergerecht installiert werden.

Buchung: Buchungen auf Sachkonten und Untersachkonten können vorgenommen werden.
Protokoll: Das während jeder Buchung mitgeführte Protokoll kann angezeigt oder ausgedruckt werden.
Stammdaten: Neuaufnahme, Änderung und Löschen von Sachkontenklassen, Sachkontengruppen, Sachkonten und Untersachkonten.
Kontoliste: Anzeige aller Sachkonten und Untersachkonten mit Kontostand und Datum der letzten Änderung; Anzeige der Gesamteinnahmen und Gesamtausgaben; Druck einer Kontoübersicht mit staatlicher Auflage, Kontostand, prozentuale Kontoauslastung, Datum der letzten Änderung, Gruppensummen und Gesamtsummen; Druck Haushaltsplan mit frei installierbarem Listenkopf, Datum, Arbeitszeitfonds, Kontenübersicht, Erfüllung der Einnahmen/Ausgaben nach Arbeitszeitfonds.
RESET: Rücksetzen der Kontostände und der staatlichen Auflage aller Konten auf den Wert null.
Der Kontomanager wurde in Turbo-Pascal 4.0 programmiert und ist auf 16-Bit-PCs (A 7150, kompatible) mit den Betriebssystemen MS-DOS oder DCP lauffähig.

Hauptauftraggeber komplexer Wohnungsbau Berlin-Pankow, Bereich Information/Applikation, Thulestraße 1, Berlin, 1100; Tel. 4700113, App. 56 oder 4728080 *Großmann*

AMBA jetzt auch auf PCs

Das Bildverarbeitungssystem AMBA steht jetzt zur Nachnutzung unter den Betriebssystemen SCP 1700 und DCP 3.20 zur Verfügung. Für folgende Hardware-Konfigurationen stehen Lösungen bereit:
- A 7100 (SCP 1700) – grafikfähig (4 Grauwerte)
- A 7150 (DCP 3.20) – grafikfähig (8 Grauwerte)
- EC 1834 mit EIP-Karte und Robotron-Grafik K 7067.15 (Mini-BVS).

Die im Rahmen einer praxisrelevanten Bildverarbeitung wichtige Bildeingabe ist für die PCs A 7100 und A 7150 prinzipiell realisierbar.

Friedrich-Schiller-Universität Jena, Sektion Technologie für den wissenschaftlichen Gerätebau, WB Digitale Bildverarbeitung, Ernst-Thälmann-Ring 32, Jena, 6900 *Prof. Dr. Voß*

Bestands- und Prüffristenüberwachung von Meßmitteln

Das Programmpaket LMM dient zur Rationalisierung der notwendigen Buchungsarbeiten bei der Ausleihe von Meßmitteln sowie zu deren Bestands- und Prüffristenüberwachung in der Meßmittelausgabe von Maschinenbaubetrieben. Es ist menügeführt und erfordert vom Benutzer keine Programmier- und Rechnerkenntnisse. Als erforderliche Hardware können alle Computer verwendet werden, die das Betriebssystem SCP oder CP/K und die Programmiersprache dBase II verarbeiten.

Das Datenbanksystem basiert auf den beiden Stammdaten Meßbereichszuordnung (jedem Meßbereich einer bestimmten Meßmittelart ist eine 3stellige Nummer zugeordnet) und Meßmittelnutzer (Name, Abteilung und 4stellige Nutzernummer). Diese Daten werden bei der Buchung vom Rechner gemeinsam mit der zusätzlich einzugebenden spezifischen Meßmittelnummer und seiner Prüffrist automatisch gekoppelt und in nach Fachgebieten getrennten Arbeitsdateien gespeichert. Mit solchen Arbeitsdateien können anzeigende Meßmittel (Fachgebiet Länge), Maßverkörperungen (Länge), Manometer, Waagen und Gewichte sowie Meßmittel weiterer Fachgebiete in ihrem Bestand und bezüglich ihrer Prüffrist leicht überwacht werden. Zur Dateneingabe in den Rechner sind jeweils nur wenige Ziffern erforderlich. Die Möglichkeiten zur buchungstechnischen Änderung der Prüffrist und des Nutzers eines Meßmittels sind gegeben. Ebenfalls lassen sich alle Meßmittel, die ein bestimmter Kollege des Betriebes nutzt, oder welche dieselbe Prüffrist haben, auflisten. Außerdem sind Mahnschreiben an Abteilungen, in denen die Prüffrist der Meßmittel nicht eingehalten wird, im Programm mit enthalten.

VEB Kühlautomat Berlin, Abteilung TBM, Segelfliegerdamm 1–45, Berlin, 1197; Tel. 6343626 (Dr. Rosahl) *Reh*

MC 80 mit LAN-Anschluß

Nachteil beim Einsatz des MC 80 ist das Fehlen moderner und schneller Massenspeicher. Deshalb wurde eine Lösung geschaffen, die Ressourcen anderer verfügbarer Rechner für diese Aufgabe zu nutzen. Als Verbindungsmöglichkeit wurde das SCOM-LAN genutzt. Mit der NANOS-NIU-Baugruppe im MC 80 ergeben sich die Verbindungsmöglichkeiten zum A 5120, PC 1715, EC 1834 usw., wenn diese Rechner auch einen SCOM-LAN-Modul enthalten. Übertragen werden 150 kBit/s. Falls für die SCOM-NIU im MC 80 kein Platz mehr frei ist, kann die Drucker-/IFSS-Karte durch die NANOS-LAN-/IFSS-Karte ersetzt werden. Weitere Voraussetzung ist, daß durch die RAM-Karten keine Waits erzeugt werden. Das vollständige Netzprogramm benutzt denselben RAM-Bereich, den sonst das Kassettenbetriebssystem belegt. Es bietet folgende Funktionen:
- Laden einer Programmdatei
- Laden einer Overlay-Datei
- Retten eines Programm-, Overlay-, Menü- oder Datenbereiches
- Ausführen des zuletzt geladenen Programms
- serielle Ausgabe von Daten bis zum Auftreten eines EOT (nutzbar für Bildschirm-Hardcopy).

Anpassungen erfolgten bisher für die Geräte LA 32/20 K mit paralleler und mit serieller Tastatur (beide mit K 2521-ZRE). Die Einbindung wird im Hause für die Gerätekonfiguration des MC 80 als Logikanalysator genutzt. Hier können das Logik- bzw. das Zustandsanalyseprogrammsystems mit den erarbeiteten Menüs geladen und Meßergebnisse in Form von Hardcopy des Bildschirms auf dem CP/M-Rechner ausgedruckt oder abgelegt werden.

Ingenieurhochschule für Seefahrt Warnemünde/Wustrow, Direktorat Wissenschaftlicher Gerätebau (Kollege Schefe), Richard-Wagner-Straße, Rostock-Warnemünde, 2530 *Prof. Dr. Mennenga*

3D-Grafik Für KC 85/3 und KC 85/4

Das Programmsystem 3D-Simulation für den KC 85/3 (siehe MP 11/1988, Seite 328) wurde hinsichtlich verarbeitbarer Datenmenge, Programmierbarkeit von Bewegungssequenzen sowie um einen komplexen Grafikeditor erweitert und für den KC 85/4 angepaßt.
- Eine Datenbankverwaltung erlaubt die verlustlose Transformation von ca. 2000 Raumpunkten pro 32-KByte-Speicherebene (bis zu 9 Ebenen gleichzeitig) oder eine parallele Textverarbeitung bzw. Basic- oder Assemblerprogrammierung.
- Für Animationen können zusätzlich zu den geradlinigen Sequenzen freie Wege für eine Betrachterbewegung im Modellraum vorprogrammiert werden.
- Ein Editor zur grafischen Bearbeitung der entwickelten 2D- oder 3D-Bilder umfaßt u. a. Linien-, Kreis-, Polygongenerierung, Flächenfüll-, Flächenschraffier-, -löschfunktionen, Sprite-, Fenster- und Vollbildverknüpfungen, Inversdarstellung, Text, Bearbeiten, Abspeichern und Laden von Ganzbildschirmgrafiken.
- Eine hochauflösende Bildgenerierung, Grafikaufbereitung und -ausgabe (bis zu 700 × 512 Pixel) erlaubt das Anfertigen von ganzseitigen Illustrationen.
- Die Handhabbarkeit des Systems wurde u. a. durch erhöhte Verarbeitungs- und Darstellungsgeschwindigkeit, erweiterte Modellgenerierung und -manipulation, Pulldown-Menüs, variable Fadenkreuz-, Gummilinien-, Gummikreis- und -fensterfunktionen verbessert.
- Die Dateien der Vorgängerversion sind übertragbar.

Fragen zum System 3D-Grafik bitte an A. Thierbach, Kräuterweg 2d, Frankfurt/Oder, 1200. Bezugsmöglichkeit besteht über:

VEB Datenverarbeitungszentrum Frankfurt/Oder, Abteilung Absatz, Faberstraße 1, Frankfurt/Oder, 1200 *Thierbach*

Grafiksystem FIGRA 3.0.

In der Mikroprozessortechnik Heft 2/88, S. 56, wurde das Grafiksystem FIGRA unter dem Betriebssystem SCP für verschiedene 8-Bit-Rechner vorgestellt. Nach einer gründlichen Überarbeitung und Erweiterung liegt jetzt die Version 3.0. von FIGRA für 8- und 16-Bit-Technik vor.
Das Anliegen von FIGRA besteht darin, dem Nutzer für die zwei- und dreidimensionale Darstellung von wissenschaftlich-technischen Ergebnissen und Zusammenhängen geeignete Prozeduren zur Verfügung zu stellen, die einfach in ihrer Handhabung sind. Dabei erfolgt eine Arbeit mit den originalen Nutzerkoordinaten, da das System automatisch eine Transformation auf ein anzugebendes Pixelraster übernimmt. FIGRA 3.0. wurde auf folgender Gerätetechnik implementiert:

1. BC A 5120 (K 1520) mit CGD-Farbgrafiksteckeinheit (ORZ Karl-Marx-Stadt) mit 256 × 256 Pixeln in 16 Farben
2. PC 1715/A 5120/A 7100 und grafikfähiger Drucker mit beliebigem Pixelraster in 2 Farben
3. A 7100 mit 640 × 400 Pixeln in 4 Helligkeitsstufen
4. A 7150 mit 640 × 480 Pixeln in 8 Farben bzw. Helligkeitsstufen
5. EC 1834/A 7150 mit 320 × 200 Pixeln in 4 Farben oder 640 × 200 Pixeln in 2 Farben
6. XT- und AT-kompatible Technik mit den Grafikkarten CGA. MCGA, EGA, VGA oder Hercules.

Die Programmierung erfolgte für 1–3 unter Turbo-Pascal 3.x und für 4–6 unter Turbo-Pascal 5.0. Durch FIGRA 3.0. werden u. a. folgende Prozeduren bereitgestellt: Festlegen des Darstellungsfeldes und des Nutzerkoordinatenbereiches, Textdarstellung in verschiedenen Größen und Richtungen, Bewegen absolut und relativ zu einem Koordinatenpunkt im Nutzerbereich, Festlegen der aktuellen Zeichenfarbe, Füllen und Umranden von Teilen des Darstellungsfeldes und von Nutzerbereichen, Hardcopy des Darstellungsfeldes, Zeichnen und Beschriften von Achsen, Darstellung von Marken, Zeichnen von Polylinien, verschiedene Prozeduren zur dreidimensionalen Darstellung mit und ohne verdeckte Linien.

Forschungsinstitut für Textiltechnologie Karl-Marx-Stadt, Abt. PM, PSF 243, Karl-Marx-Stadt, 9010; Tel. 5709436 oder 435 *Dr. Gebhardt*

Wir suchen ...

... eine Möglichkeit, die Schreibmaschine S 6001 als Drucker für den KC 85/3 nutzen zu können.
Berufsschule „Thomas Müntzer", Karl-Liebknecht-Straße, Blankenhain, 5304; Tel. 2269 *Heyner*

... ein Computerprogramm – lauffähig auf A 7100 (unter SCP) zur Nutzensermittlung für Neuerungen speziell im Bauwesen, aber nicht Bedingung.

VEB Bau Grimmen, Produktionsbereich Berlin, Koll. Seidel, Rigaer Straße 27, Berlin, 1035 *Scheufler/Seidel*

... nach einer technischen Lösung (Treiber) für das Betreiben eines Digitalisiertabletts K 6405 an einem Schneider-PC 1512/1640 zur rechnergestützten Entwurfsarbeit mit dem Grafikprogrammpaket CADdy.

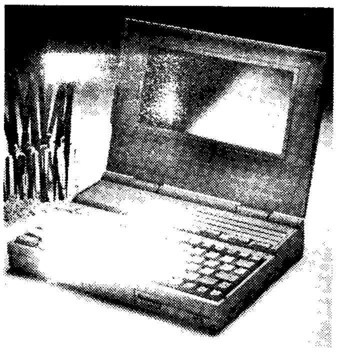
VEB Kombinat Rationalisierung Suhl, Stammbetrieb, Industriegelände, PSF 403, Abt. ITZ, Suhl, 6019; Tel. 4782/App. 38 *Peter*

Intels 80960-Familie wächst

Bereits im Frühjahr 1988 kündigte Intel die ersten 32-Bit-Einchipmikrorechner an: die Familie 80960 und den 80376. Letzterer ist ein aus dem 80386 entwickelter Controller, der etwa dreimal so leistungsstark wie der 16-Bit-Controller 80186 ist und der speziell auf die Belange der Computerperipherie, der Nachrichtentechnik und der Datenkommunikation zugeschnitten ist. Die Architektur des 80960 mit seinen rund 350 000 Transistoren dagegen ist das Basiskonzept für eine Vielzahl künftiger hochspezialisierter Controller-Varianten. Vorrangige Einsatzbereiche können die Bildverarbeitung, Protokoll-Handling und Motorsteuerungen sein. Durch die Anwendung von RISC- und Parallelverarbeitungsmethoden konnte ein Leistungsniveau erzielt werden, das weit über dem herkömmlicher Mikroprozessoren liegt. Die ersten Typen waren der 80960 KA, der 80960 KB und der 80960 MC. Der KB ist ein vollständiges Single-Chip-CPU-Subsystem mit einer Rechnerleistung von 7 bis 10 MIPS bei einer Taktfrequenz von 20 MHz. Zu seinen Besonderheiten gehören Boolesche Befehle, Bitbefehle, Selbsttest- und Debug-Funktionen. Er enthält einen 512-Byte-Befehlscache, einen Registercache, einen 32-Bit-Multiplex-Burstbus und einen Interruptcontroller mit 256 Vektoren und 32 Prioritätsebenen. Der KA ist ein KB ohne Gleitkommaarithmetik. Der MC bietet unter anderem zusätzlich ein Speichermanagement mit Paging, String-Befehle und zusammen mit dem 82965 die Unterstützung von drei verschiedenen Fehlertoleranzstufen.
Ende vorigen Jahres kündigte Intel nun die Versionen XA und CA des 80960 an. Der XA (Extended Architecture) ermöglicht mit Hilfe eines 33. Bits (Tag-Bit) sowohl Objektadressierung als auch -schutz durch das Betriebssystem. Der CA dagegen soll pro Taktzyklus gleich zwei bis drei Maschinenbefehle abarbeiten können; typisch für RISC-Systeme war bisher, daß pro Taktzyklus ein Maschinenbefehl abgearbeitet wird. Damit soll er eine maximale Leistung von 66 MIPS ermöglichen. Die Familie 80960 werden wir zu einem späteren Zeitpunkt ausführlicher vorstellen (siehe auch MP 8/1989, Seite 251).
MP

Neu von Compaq: LTE und LTE/286

Auf dem Weg zu immer leistungsfähigeren transportablen Personalcomputern (PCs) schreitet auch die Firma Compaq zügig voran. Nach der Vorstellung ihres ersten battariebetriebenen PCs, des SLT/286, im Oktober 1988 stellte sie vor drei Monaten als Ergänzung zwei weitere Modelle vor. Sie unterscheiden sich zwar kaum äußerlich, jedoch besitzt der LTE/286 gegenüber dem LTE nicht nur einen leistungsfähigeren Prozessor (LTE: Intel 80C86 mit 4,77 bzw. 9,54 MHz, LTE/286: Intel 80C286 mit 8 bzw. 12 MHz, sondern auch umfang-



reichere Speicherkapazitäten sowie einen Sockel für den mathematischen Koprozessor 80C287. Die LTEs werden von Compaq als die ersten Computer im Notizblockformat (allerdings mit $27{,}9 \times 21{,}6\,cm^2$ Grundfläche) mit dem vollen Funktionsumfang eines PCs bezeichnet. So sollen trotz der niedrigen Bauhöhe von 4,8 cm keine Abstriche bezüglich der grundlegenden Bestandteile und der Funktionalität eines standardmäßigen PCs bestehen. Die Merkmale sind: ein 40- oder 20-Megabyte-Festplattenlaufwerk mit einer Bauhöhe von nur 1,9 cm, mit einer mittleren Zugriffszeit von unter 29 ms und einem Interleave-Faktor von 1 : 1, ein ebenfalls nur 1,9 cm hohes 3,5-Zoll-Diskettenlaufwerk, das den Einsatz von Standard-Software und den Datenaustausch mit anderen PCs ermöglicht; eine 1,25 cm hohe Tastatur mit 81 Tasten, die kompatibel zu Anwendungen ist, die für Tastaturen mit 101 Tasten geschrieben wurden; und schließlich ein nur 0,64 cm dünner, CGA-kompatibler Elektrolumineszenz-Bildschirm mit einer Auflösung von 640 × 200 Punkten bzw. 25 Zeilen zu 80 Zeichen. Der 640-KByte-RAM der 2,8 kg schweren PCs läßt sich auf 1 MByte (LTE) bzw. 2,6 MByte (LTE/286) erweitern. Im Lieferumfang ist der Compaq Expanded Memory Manager (CEMMP) enthalten, der die LIM/EMS-Spezifikation 4.0 und damit den Zugriff auf den RAM-Bereich über 640 KByte unterstützt.
Mit dem standardmäßigen NiCd-Akkublock werden etwa 3,5 Stunden Betriebsdauer erreicht. *MP*

Mac IIci: Apples neues Flaggschiff

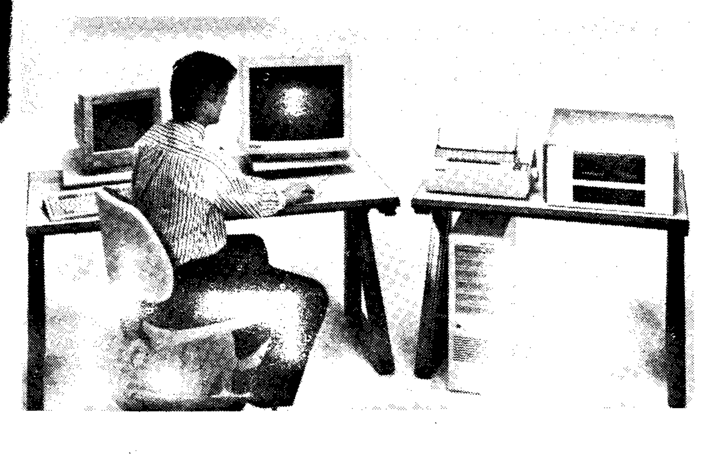
Gleichzeitig mit ihrem ersten Macintosh-Laptop, über den wir demnächst berichten werden, stellte die Firma Apple im September vergangenen Jahres das neue Flaggschiff der Macintosh-Familie vor. Der **Macintosh IIci** gleicht im Aufbau dem Mac IIcx, besitzt jedoch statt des Prozessors Motorola 68030 mit 16 MHz die Version mit 25 MHz. Die gegenüber dem cx mit 45 Prozent angegebene Leistungssteigerung des ci (c steht für compact, i für increased performance – gesteigerte Leistung) beruht jedoch nicht nur auf der höheren Taktfrequenz. Wurden bisher nur 120-Nanosekunden-RAMs verwendet, so erlauben die 80-Nanosekunden-RAMs mit Fast-page-Modus im ci die Burstfill-Betriebsart des Prozessors. Zur Beschleunigung komplexer mathematischer Berechnungen wird bereits standardmäßig der 25-MHz-Koprozessor Motorola 68882 eingesetzt. Wird eine Cache-Zusatzkarte verwendet, läßt sich die Leistung um weitere 20 bis 30 Prozent steigern. Der 1-MByte-Hauptspeicher kann mit 256-KByte-Modulen (SIMMs) auf der Leiterkarte bis zu 8 MByte ausgebaut werden; mit 4-MByte-Modulen wären 32 MByte möglich. Der ROM wurde gegenüber dem cx auf 512 KByte verdoppelt. Erwähnenswert ist auch, daß die Hauptplatine des ci erstmals komplett in SMD-Technik und in 4 Lagen aufgebaut ist. Außerdem soll die Leiterkarte robuster gegenüber Störeinwirkungen sein.
Lange wird der ci jedoch scheinbar nicht das Macintosh-Flaggschiff bleiben, im Gespräch ist bereits der Mac IIcr mit 33-MHz-CPU und sechs 20-MHz-NuBus-Slots, der bis zu viermal schneller werden soll als der Mac IIci.
MP

Commodores neue Amiga-Systemfamilie

Der Amiga 2500/20, als Basisgerät des Commodore-DTP-Paketes seit Mitte 1989 am Markt, ist jetzt als Einzelgerät verfügbar. Dieses Grundsystem bildet die Basis für alle professionellen Anwendungen wie Desktop Publishing (DTP), Desktop-Video, Desktop Presentation und andere Arbeitsgebiete. Amiga-2000-Besitzer können ihre Geräte mit der entsprechenden Commodore-Zusatzkarte problemlos zum leistungsstärkeren 2500er aufrüsten. Der Einstieg ist also auch in Zukunft mit dem preisgünstigen Amiga 2000 möglich; bei steigenden Ansprüchen braucht lediglich die Zusatzkarte nachgekauft und in einen der Slots gesteckt zu werden.
Der 2500/20 arbeitet mit dem 14,2 MHz-Motorola-Prozessor 68020, dem mathematischen Koprozessor 68881 und den Amiga-spezifischen Koprozessoren für Datenverwaltung, Video, Grafik und Ton. Standardmäßig stehen 3 MByte Arbeitsspeicher und eine Festplatte von 40 MByte mit Autoboot-Controller zur Verfügung. Ein Laufwerk für 3,5-Zoll-Disketten ist eingebaut, ein weiterer freier Geräteschacht für Festplatten oder Diskettenlaufwerke (5,25 oder 3,5 Zoll) ist vorhanden. Bei 3 freien Steckplätzen für Amiga-Karten, 4 PC- bzw. AT-kompatiblen Steckplätzen und einem Video-Slot für PAL-Videokarten oder Genlock wird auch der Anschluß an die MS-DOS-Welt berücksichtigt – wichtig für Anwender, die MS-DOS-Programme und -Daten weiter nutzen wollen. Das Steckplatz- und Bussystem macht aber auch den Ausbau der Amiga-Professional-Familie bis hin zur Workstation möglich. Die beiden Bussysteme des Amiga sind mit einer Brückenkarte miteinander verbunden; dies erlaubt die Datenübergabe zwischen verschiedenen Betriebssystemen unabhängig von den eingesetzten Prozessoren. Dazu kommen die Multitaskingfähigkeiten. Während beispielsweise am Bildschirm Texte bearbeitet werden, können in einer Datenbank parallel dazu bereits die Adressen selektiert werden. Mit der offenen Systemarchitektur und seiner Anpassungsfähigkeit sieht Commodore den Amiga 2500 für die Zukunft gerüstet – gleich, ob der Zug für den Anwender auf das MS-DOS-Gleis oder in UNIX-Richtung fährt. *MP*

Workstations von Du Pont

Der Chemiegigant Du Pont steigt nun auch in den Workstationmarkt ein.
Auf der SYSTEMS 89 stellte die Firma eine Unix-Workstation mit einer Leistung von 13 MIPS vor, deren Herz eine VMEbus-Doppel-Europaformatleiterkarte mit dem RISC-Mikroprozessorsatz Clipper 300 ist – bRISC genannt.
MacbRISC heißt dagegen eine NuBus-Leiterkarte, mit der ein Apple Macintosh II zu einer Workstation aufgerüstet werden kann.
Von der Du Pont de Nemours (Deutschland) GmbH wurde zur Productronica 89 die PC-Workstation SeriCAM (s. Bild) vorgestellt, die besonders dem kleineren und mittleren Hersteller gedruckter Schaltungen einen preisgünstigen Einstieg in die digitale Panel-Produktion ermöglichen soll. Gestützt auf 386er Hardware, wie HP-Vectra RS 20 und Compaq-PCs, entwickelt das Unternehmen eine Software, die Daten aller gängigen CAD-Systeme verarbeiten kann. SeriCAM schließt die Lücke zwischen Design und Herstellung gedruckter Schaltungen. Qualitätsverluste, wie sie durch Reproduktionsschritte entstehen können, sind durch die digitale Verarbeitung der Leiterplatten-Layouts praktisch ausgeschlossen. Statt in zeitraubenden manuellen Arbeitsschritten entstehen die Panels und Produktionsdaten elektronisch auf der Workstation. Zum Lieferumfang gehören ferner Programme zur Entwicklung von Test-Coupons und zur automatischen Erstellung der Lötstopmasken. Auch Editierfunktionen zur Korrektur von Fehlern, die ein eingebautes Design-Check-Programm erkennt, unterstützen die Arbeit. Schließlich liefert SeriCAM Daten für NC-gesteuerte Bohr- und Fräsmaschinen, womit die Integration in die automatisierte Fertigung, und damit ein erster Schritt zum computer-integrated manufacturing (CIM), möglich wird. Bei der Gestaltung der Benutzeroberfläche wurde bewußt auf eine einfache Menüsteuerung Wert gelegt, so daß der Bediener nicht über weitreichende Programmierkenntnisse verfügen muß. Der Einsatz von SeriCAM, dem ersten Gerät eines ganzen Systems elektronischer Produktionssysteme von Du Pont, erlaubt vor allem die wirtschaftliche Herstellung von Kleinserien gedruckter Schaltungen bis hinab zur Losgröße 1. Anders als bisherige PC-Systeme arbeitet SeriCAM – nach Ver-



lautbarungen von Du Pont – beispielsweise mit etwa 80 Prozent der Leistung einer Unix-Workstation, die sich durch ihren höheren Preis eher für Großserien bezahlt macht. *MP*

Clipper 5.0

Der Langersehnte ist nun da: der bekannte dBase-Compiler Clipper in einer neuen Version. Wurde die letzte Version von Nantucket noch mit Sommer '87 bezeichnet, so blieb die Kölner Softwarefirma diesem Trend nicht treu, vielleicht mit Herbst '89, sondern paßte sich dem gegenwärtigen Stand an und nannte ihr neues Produkt gleich Version 5.0; ähnlich Ashton-Tate, Borland und Microsoft, die sich ja auch schon in der vierten bzw. fünften Ebene ihrer aktuellen Versionen von dBase, Turbo-Pascal bzw. MS-DOS bewegen. Sind vielleicht nicht nur die Norton Guides, sondern auch die Versionsnummer von einem anderen Software-Haus übernommen, so daß der Preis mit etwa 2 700 DM mehr als fünfmal so hoch ist wie der für Turbo-Pascal 5.0? Tatsache ist, daß viele Neuerungen die Clipper-Fans begeistern werden. Um vieles kümmern sich jetzt Compiler und Linker; beispielsweise ist das Programmieren in Overlaytechnik nicht mehr erforderlich. Der speziell für Clipper 5.0 nach neuen Konzepten entwickelte Linker arbeitet wesentlich schneller, wodurch die Zeiten für die Programmentwicklung deutlich verkürzt werden. Die 640-KByte-Grenze von MS-DOS wurde übersprungen, und eine Clipper-Bibliothek bietet die Möglichkeit der Online-Hilfe in Anwendungsprogrammen, in die auch die Norton Guides bereits integriert sind. Des weiteren ist die Deklaration mehrdimensionaler Felder (Arrays) möglich. Nachdem Nantucket angekündigt hatte, mit neuen Versionen den dBase-Pfad zu verlassen und auf eigenen Wegen zu clippern, müssen aber Tests nachweisen, inwieweit auch weiterhin Daten und Programme von dBase bzw. der Version Sommer '87 verwendet werden können. *MP*

Neue Möglichkeiten für Flüssigkristall-Anzeigen

Ferroelektrische Flüssigkristall-Anzeigen könnten aufgrund ihrer Schaltzeiten unter 10 Mikrosekunden, ihrer hohen spontanen Polarisation und des bistabilen Verhaltens der Flüssigkristalle im spannungslosen Zustand die geeignete Technologie der neuen Generation sein. Damit könnten Flüssigkristall-Anzeigen auch für die Darstellung bewegter, hochauflösender Bilder eingesetzt werden.
Obwohl die Eignung der ferroelektrischen Flüssigkristall-Anzeigen nachgewiesen wurde, sind bis zu ihrer industriellen Anwendung noch fertigungstechnische Schwierigkeiten zu überwinden. Sie liegen in der geeigneten Behandlung der Kristalloberflächen. Die sogenannten snektischen Schichten müssen senkrecht zu den transparenten Elektroden der Flüssigkristall-Zellen ausgerichtet sein, und die Flüssigkristall-Schichten dürfen nur ein bis zwei Mikrometer dick sein, damit störende Materialeffekte der Kristalle unterdrückt werden können.

Quelle: Blick durch die Wirtschaft vom 11. 8. 1989 *Wi*

Platinenreinigung ohne FCKW

Die japanischen Firmen Mitsubishi und Tiayo Sanso haben eine besonders für den industriellen Einsatz geeignete Reinigungsmaschine entwikkelt, die ohne umweltbelastende Flourchlorkohlenwasserstoffe auskommen soll.
Während des Reinigungsvorganges werden Fremdpartikel bis zu 0,1 μm durch einen Eisfilm gebunden, der, einschließlich der Partikel, mit flüssigem Stickstoff entfernt wird.
Diese Maschine ist für den Einsatz in der Halbleiter- und Platinenfertigung vorgesehen und soll Anfang 1990 auf den Markt kommen.

Quelle: Elektronik. – München 38 (1989) 14. – S. 7 *Fa*

Joint-Venture für PC-Produktion

Wie die PC-Woche in ihrer Ausgabe 41/89 berichtet, haben die sowjetische Wissenschafts- und Produktionsvereinigung Sojusutschpribor und die US-amerikanische Phoenix Group International ein gemischtes Unternehmen zur Fertigung von Personalcomputern in der UdSSR gegründet. Das **Samcon** genannte Unternehmen soll bereits 1990 11 000 Einheiten für sowjetische Schulen und Hochschulen herstellen; bis 1995 ist die schrittweise Produktionssteigerung auf bis zu 150 000 Einheiten geplant. *MP*

Einschätzung zur Entwicklung neuraler Netze

Obwohl bereits sehr viel über neurale Computer veröffentlicht wird, sind praktische Lösungen bisher nur in geringem Umfang verfügbar. Neurale Computer sollen nicht nur schneller arbeiten, sie sollen auch lernen können. Gegenwärtig werden nur Geräte angeboten, die einfache Formen erkennen können. Die Firma SRI International rechnet damit, daß im Jahre 1992 Kontrollsysteme für Roboter und Worterkennungssysteme verfügbar sein werden. „Sehen“ könnten Maschinen bis 1993 gelernt haben. Mit der Verfügbarkeit optischer Speicher, schneller Gallium-Arsenid-Chips und optischer Chips im Zeitraum 1994/1995 sind komplexe Anwendungen möglich. Bis zum Jahre 1999 soll der Umsatz lernender Systeme im Vergleich zum gegenwärtigen Stand um das Vierfache anwachsen.

Quelle: Handelsblatt vom 15. 8. 1989 *Wi*

Hochintegrierte Logikchips

Hochintegrierte Chips, die bis zu 800 000 CMOS-Transistoren und bis zu 60 000 aktive logische Schaltkreise enthalten, wurden im IBM-Labor in Böblingen entwickelt. Diese Chips, die nur 12,7 × 12,7 mm^2 groß sind, werden bereits in den neuen Modellen der IBM-Rechnerfamilie ES/9370 eingesetzt. Die Schaltkreismuster auf diesen Chips haben eine Breite von nur einem tausendstel Millimeter. Die Systemlogik und die Speichermakros wurden mit Hilfe einer Schaltkreis-Bibliothek (logische Grundschaltkreise) erzeugt. Trotz der höheren Integration wird bei den neuen Chips eine geringere elektrische Verlustleistung frei als bei den herkömmlichen Chips. Die gesamte Architektur des Systems 9370 kann aufgrund dieser dicht bepackten Logikchips nicht, wie bisher, auf vier Prozessorkarten, sondern auf einer einzigen angeordnet werden. Auf dieser Prozessorkarte sind sieben neue Chips untergebracht: ein 32-Bit-Mikroprozessor, ein Cache-Controller, ein Hauptspeicher-Controller, zwei Gleitkomma-Koprozessoren, ein Taktgeber-Chip und ein Bus-Adapter-Chip.

Quelle: Neue Zürcher Zeitung vom 31. 5. 1989 *Fa*

Nickel-Hydrid-Akkumulatoren

Neben mehreren Forschungsinstituten arbeiten auch Batteriehersteller in den USA, in Japan und in Westeuropa an der Entwicklung von Kleinakkumulatoren, speziell an der Entwicklung einer Nickel-Hydrid-Zelle.
Während der chemische Prozeß seit langem bekannt ist, gab es mit der praktischen Realisierung bisher Probleme. Ältere Versuchsmuster wiesen eine viel zu kurze Lebensdauer auf. Das Anodenmetall dehnte sich bei der Absorption des Wasserstoffs aus und zog sich bei der Abgabe wieder zusammen. Dadurch versprödete es und zerkrümelte.
An der Tokai-Universität in Japan wurde das Metall (Legierung aus Lanthan, Nickel und Aluminium) als Pulver in einen Träger aus einem Silikon-Polymer eingebettet und zu einer etwa 0,3 mm dicken Folie ausgewalzt. Damit aufgebaute Zellen hielten 1 000 Lade-Entlade-Zyklen stand. Die Energiedichte erreichte etwa den doppelten Wert des Ni/Cd-Akkus.
Eine ähnliche Zelle entwickelten die Firmen Varta und Ovonic. Die Zylinderzelle ist 49 mm hoch und hat einen Durchmesser von 26 mm. Sie liefert eine Spannung von 1,3 V, weist eine Nennkapazität von 3 Ah auf (Entladung mit 0,2 CA bis 1 V) und gestattet Hochstromentnahme bis zu 5 CA. Sie ist innerhalb einer Stunde schnellladbar und bei Ladung mit 0,1 CA überladesicher. Am Ende der Entladung tritt kein plötzlicher Spannungseinbruch auf. Nach 1 000 Lade-Entlade-Zyklen ist die Kapazität erst auf 60 % des Nennwertes abgesunken.
Auch die französische Firma SAFT arbeitet an der Entwicklung einer Nikkel-Hydrid-Zelle. Es wurden bereits Versuchsmuster für die Energieversorgung und für Kommunikationssatelliten und Orbitalstationen der ESA (European Space Agency) geprüft und für gut befunden.
Die Nickel-Hydrid-Zelle befindet sich noch im Teststadium. In den kommenden Jahren sollen Muster den Geräteherstellern zu Versuchszwekken zur Verfügung gestellt werden. Die Einführung für den Endverbraucher wird im Laufe der 90er Jahre erwartet.

Quelle: Elektronik. – München 38 (1989) 14. – S. 28 *Fa*

Chip für die Bildeinblendung in Fernsehgeräten

Das Einblenden eines verkleinerten Bildes in ein Fernsehbild ist schon seit einigen Jahren bekannt. Es konnte sich jedoch aufgrund der hohen Kosten bei der analogen Bilddarstellung nicht durchsetzen. Besonders der notwendige digitale Bildspeicher und der Digital-Analog-Wandler verursachten einen hohen Schaltungsaufwand.
Die Siemens AG hat einen Chip entwickelt, der alle erforderlichen Einheiten, wie Bildspeicher, Wandler und Signalprozessor, enthält. Dieser Chip erlaubt die Verkleinerung des Bildes und die Synchronisation mit dem Hauptbild. Der Chip ist so konzipiert, daß er sich an die gerätespezifische Geometrie und an die verschiedenen Fernsehstandards (NTSC, PAL, Secam) anpassen läßt. Die Möglichkeiten der neuen Fernsehgeneration, die mit Bildfrequenzen von 100 Hz arbeitet, wurde bei der Entwicklung berücksichtigt. Der Bild-im-Bild-Prozessor kann sowohl in digitaler, gemischt analog/digitaler und analoger Schaltungsumgebung betrieben werden. Auf dem Chip werden rund 350 000 Transistoren integriert. Über 40 Prozent der Entwicklungskosten entfielen auf die Softwareentwicklung für den Schaltkreisentwurf und deren Simulation. Die westeuropäischen und die japanischen Fernsehgerätehersteller haben bereits Geräte mit dem neuen Chip im Angebot.

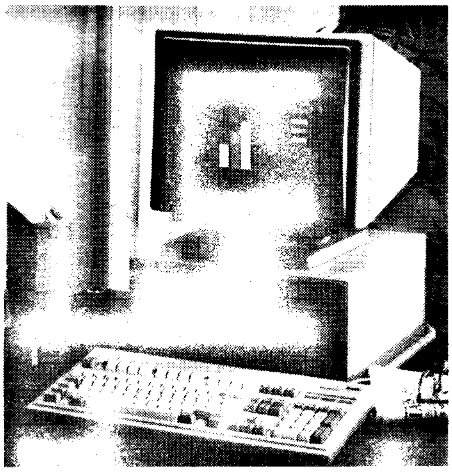
Quelle: Blick durch die Wirtschaft vom 17. 7. 1989 *Wi*

vorgestellt

Industriestandard-PCs von DEC

Das Einstiegsmodell der DECstation-PC-Serie, DECstation 200, basierend auf dem 16-Bit-Prozessor Intel 80286

Den mittleren Leistungsbereich mit rechenintensiven Anwendungen deckt die DECstation 300 mit dem 16/32-Bit-Prozessor Intel 80386SX ab



Die DECstation 350 als Spitzenmodell der DECstation-PC-Familie verfügt über den mit 20 MHz getakteten 32-Bit-Mikroprozessor Intel 80386

Die im ersten Halbjahr für den USA-Markt angekündigten Industriestandard-PCs von Digital Equipment Corporation (DEC) (siehe MP 10/89) werden nun seit September 1989 auch auf dem europäischen Markt angeboten. Die Personalcomputer DECstation 200, 300 und 350 basieren auf den Mikroprozessoren Intel 80286, 80386SX bzw. 80386 und sind zum Industriestandard, der durch die IBM-PCs geprägt wurde, kompatibel.
Die DECstations sind standardmäßig mit dem VGA-Bildschirmadapter (Video Graphics Array) ausgerüstet, der eine Auflösung von 640 × 480 Pixel gewährleistet. Es steht ein monochromatischer Monitor (12″, 64 Graustufen) oder ein Farbmonitor (14″, 256 Farben aus 262144 Farben) zur Verfügung. Die Bildschirmwiederholrate beträgt 70 Hz.
Die grundsätzlichen Eigenschaften und Parameter der drei DECstations sind aus nebenstehender Tabelle ersichtlich. Darüber hinaus stehen noch eine Reihe von Systemerweiterungen wie Arithmetik-Coprozessoren, Hauptspeichererweiterungen, Magnetbandsubsysteme, eine breite Palette serieller und paralleler Tischdrucker sowie die Ethernet-Steuereinheit DEPCA zur Verfügung, über die auch eine Maus mit zwei Knöpfen bedient wird. Durch den Ethernet-Anschluß können die DECstations in PC-Netze (PCLAN)auf der Grundlage der Personal Computing Systems Architecture (PCSA) integriert werden.
Als Server können hier die bewährten VAX- und RISC-basierenden Workstations, Mehrbenutzer- und Serversysteme wie DECstation 5400, DECstation 5810/5820, VAXsystem 3300/3400, VAXsystem 3800/3900 oder Anlagen der VAX-6000-Serie verwendet werden.
Auf den DECstation-PCs laufen alle heute verfügbaren MS-DOS-Anwendungen. Außerdem sind sie offen für bereits erhältliche und zukünftige Anwendungssoftware unter den Betriebssystemen MS-OS/2, Unix, Xenix und anderen. Für das MS-DOS wird die DECnet/OSI-Software angeboten. Für UNIX-kompatible Betriebssysteme steht auch TCP/IP zur Verfügung.
Alle Modelle werden mit zwölfmonatiger Gewährleistung ausgeliefert, der sich ein umfassendes Angebot an Serviceleistungen von DEC anschließt. Der Vertrieb der neuen PCs erfolgt vorrangig durch autorisierte Distributoren, wie z. B. ELECTRONIC 2000.

Prof. Dr. Th. Horn

Mikroprozessor	Intel 80286		Intel 80386SX	Intel 80386
Taktfrequenz	8 MHz		16 MHz	20 MHz
Bussystem	16-Bit-Standard-Bus		16-Bit-Standard-Bus	32-Bit-Bus
Hauptspeicher				
Standard	1 MByte		2 MByte	2 MByte
Maximal	16 MByte		14 MByte	14 MByte
Ethernetanschluß	optional		optional	optional
Diskettenlaufwerke				
3 1/2″ (1,44 MByte)	2	1	1	1
5 1/4″ (1,2 MByte)	–	–	(1)	(1)
Winchesterlaufwerke				
20 MByte	–	1	–	–
40 MByte	(1)	–	1	–
80 MByte	–	–	–	1
Grafik	VGA		VGA	VGA
Auflösung (in Pixel)	640 × 480		640 × 480	640 × 480
Bildschirm	12″ monochrom oder 14″ color		12″ monochrom oder 14″ color	12″ monochrom oder 14″ color
freie XT/AT-Steckplätze	3		5	4
Preis	ab 6 300 DM		ab 9 900 DM	ab 16 100 DM

TERMINE

Fachtagung „Information – Kommunikation – Innovation“

WER? Bezirksvorstand Berlin der KDT, AG(B) Information/Dokumentation
WANN? 23. und 24. Mai 1990
WO? Berlin
WAS?
- Austausch von Angeboten und Erfahrungen zur optimalen Informationsversorgung
- Informations- und Kommunikationstechnologien zur Produkt- und Verfahrensinnovation
- Softwarelösungen und Informationstechnik zur Sicherung eines hohen Informationsstandes von Endnutzern

WIE? Bitte Veranstaltungsprogramm und Teilnahmemeldungen anfordern bei der Kammer der Technik, Bezirksvorstand Berlin, Bereich Wissenschaft und Technik, Kolln. Juchert, Kronenstraße 18, Berlin, 1080; Tel. 2 00 09 66, App. 75

Dr. Lange

ISSN 0232-2892
Mikroprozessortechnik, Berlin 4 (1990) 1
S. 1–32

vorgestellt

Atari PC 4

Fotos: Becker

Erst in den letzten Jahren hielt Atari mit seiner PC-Linie Einzug in die Welt der MS-DOS-Rechner. Atari ging damit in die Offensive: Man wollte auch in dieser Rechner-Kategorie die Philosophie einer preiswerten Spitzentechnologie etablieren. Die Resonanz am Markt lag auch prompt über den zunächst vorsichtigen Einschätzungen des Entwicklers. Nach diesem Start stellte das Unternehmen im vergangenen Jahr den PC 4 vor. Er ist voll AT-kompatibel und basiert auf einem Intel-80286-Mikroprozessor, der softwaremäßig von acht Megahertz auf zwölf Megahertz Taktfrequenz umgeschaltet werden kann. Bei allen Taktfrequenzen arbeitet die CPU ohne Wartezyklen (Waitstates). (Eine 16-MHz-Version ist übrigens geplant.)

Der PC 4 verfügt über einen Sokkel für die Aufnahme eines arithmetischen Koprozessors 80287 und ermöglicht so die Unterstützung des Zentralprozessors im Bedarfsfall.

In seiner Grundkonfiguration ist der PC 4 mit 1 MByte RAM ausgestattet, der sich auf dem Motherboard mittels SIMMs bis zu 8 MByte aufrüsten läßt. Für diesen vollen Ausbau sind allerdings Megabytemodule Voraussetzung; mit 256-KByte-Modulen sind 5 MByte möglich. Der ROM beträgt 96 KByte.

Hinsichtlich der Massenspeicher stehen verschiedene Varianten zur Auswahl. Allen gemeinsam ist ein Diskettenlaufwerk – entweder ein 5,25-Zoll-Modell mit 1,2 MByte oder ein 3,5-Zoll-Modell mit 1,44 MByte. Als weiteres Laufwerk kann zwischen einer im allgemeinen üblichen Harddisk mit 60 MByte und einer 44 MByte fassenden Wechselfestplatte gewählt werden. Das von der Firma Syquest produzierte Wechselplattenlaufwerk im 5,25-Zoll-Slimlineformat arbeitet mit Wechselplatten, die in einem stabilen Plastgehäuse untergebracht sind (Cartridges) und sich damit wie Disketten handhaben lassen (siehe unteres Bild). Sie vereinen damit den Vorzug der Wechsel- und Sammelmöglichkeit der Disketten mit der bei Festplatten üblichen höheren Kapazität. Zudem stellt die mittlere Zugriffszeit von 39,6 ms einen akzeptablen Wert für ein Wechselmedium dar (die Datenübertragungsrate beträgt etwa 500 KByte/s). Zur Zuverlässigkeit dieses Mediums liegen noch keine Langzeiterfahrungen vor; vom Hersteller wird für das Laufwerk eine Lebensdauer von 20 000 Stunden angegeben, für die Cartridge von 10 000 Wechselvorgängen.

An den PC 4 lassen sich die gängigen Monochrom- und Farbmonitore anschließen (sowohl analog als auch TTL). Auf der Mutterplatine ist bereits eine Grafikkarte installiert, die die bekannten Standards Hercules, CGA, MDA, EGA und VGA unterstützt, so daß ein Steckplatz gewonnen wird. Bei CGA ist beispielsweise eine Auflösung von 640 × 200, bei EGA von 640 × 350 und bei VGA von 720 × 348 Pixeln möglich. Bei Bedarf läßt sich diese Grafikkarte jedoch auch abschalten und eine Zusatzkarte verwenden. Serienmäßig wird der PC 4 mit dem Monochrommonitor von Atari, dem PCM 124, ausgestattet. Der nachleuchtende 14-Zoll-Bildschirm ist besonders für Textanwendungen gedacht; wer häufig farbige Umsetzungen benötigt, sollte den Erwerb eines hochauflösenden Farbmonitors in Erwägung ziehen.

Auch die parallele und die beiden seriellen Schnittstellen sowie die Disk-Controller sind auf der Mutterplatine installiert, was sich ebenfalls positiv auf den Platz für Erweiterungskarten auswirkt.

Insgesamt fünf Slots ermöglichen somit die Verwendung zusätzlicher AT-Karten für die unterschiedlichen Anwendungsmöglichkeiten. Hierbei bietet das Gehäuse des PC 4 nicht nur den langen, sondern auch den hohen Steckkarten ausreichend Platz. Von den fünf Steckplätzen sind vier auf die Aufnahme von AT-Karten ausgelegt. Der fünfte Slot dient der Aufnahme einer XT-üblichen Erweiterungskarte.

Geliefert wird der PC 4 mit der notwendigen Software: MS-DOS 3.3, Basic-Interpreter GW-Basic, MS-Windows, zwei Utility-Disketten für die Wechselplatte und die VGA-Grafik. Ebenfalls zum Lieferumfang gehört eine Maus.

MP-We

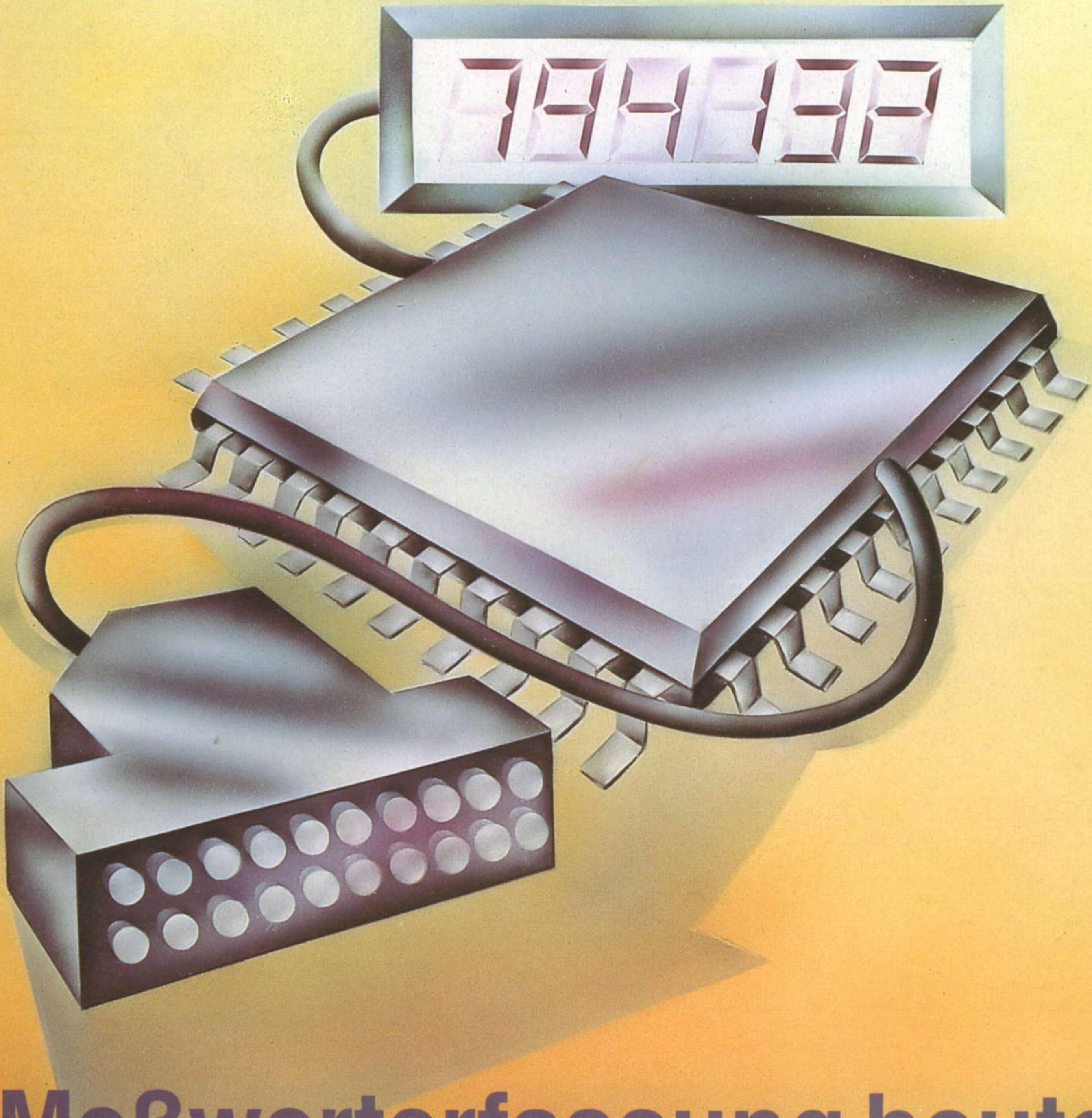
ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

MP

Heft 2 · 1990

Mikroprozessortechnik

VEB Verlag Technik Berlin
ISSN 0232-2892

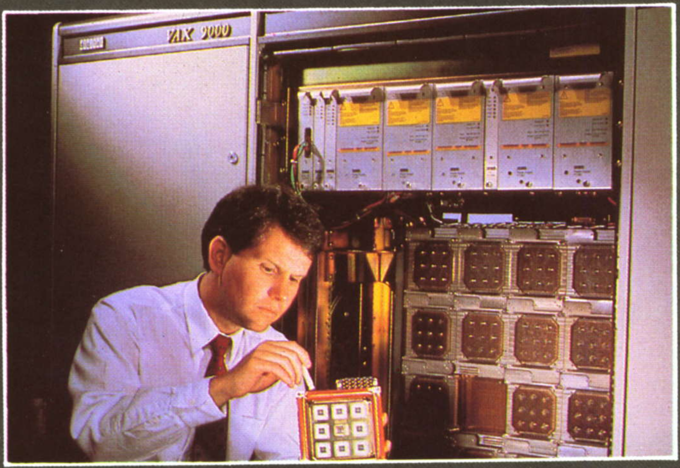


Meßwerterfassung heute

Neue Rechnerfamilie VAX 9000 vorgestellt

Mit der neuen Computerfamilie VAX 9000 will DEC nun auch in den Markt der Großrechner eindringen.

Als technischen Durchbruch in der Halbleiterfertigung sieht DEC die neue Art der Multi-Chip-Packung, mit der die Leitungswege so verkürzt werden, daß im Vergleich zur konventionellen Leiterplattentechnologie eine Verdoppelung der Computerleistung erzielt werden kann. Es ergeben sich bis zu 30mal höhere Packungsdichten. In der neuen Großrechnerfamilie VAX 9000 wird diese Technologie erstmals eingesetzt (Bild unten).



Die Fa. Digital Equipment Corporation (DEC) hat Ende Oktober 1989 eine neue Familie von VAX-Rechnern, die Serie 9000, vorgestellt. Die Systeme sind die bisher leistungsfähigsten des Unternehmens und erreichen je nach Anzahl der Prozessoren die 30- bis 117fache Leistung einer VAX-11/780, des weltweit ersten 32-Bit-Kleinrechners.
Ein Prozessorelement hat somit eine Leistung von über 30 MIPS (Millionen Befehle pro Sekunde). Neu ist auch, daß jedes Prozessorelement noch über einen Vektorprozessor verfügt, der eine Verarbeitungsleistung von 125 MFLOPS (Millionen Gleitkommaoperationen pro Sekunde) erreicht. Maximal kann das System aus vier Prozessorelementen mit vier Vektorprozessoren bestehen.
Der Hauptspeicher, in dem 1-MBit-Chips eingesetzt werden, kann aus zwei Moduln zu je 256 MByte bestehen. Mit dem Übergang auf den 4-MBit-Chip erweitert sich die Kapazität des physischen Hauptspeichers auf 2 GByte.
Das Ein- und Ausgabesystem besteht aus max. 4 Hochleistungsbussystemen XMI (ca. 100 MByte/s). Der XMI-Bus war mit der Serie VAX 6000 als rechnerinterner Hauptbus bereits vor ca. 2 Jahren eingeführt worden. Bei vier Bussystemen beträgt die maximale Datentransferrate 320 MByte/s. An den XMI können analog der VAX 6000 VAXBI-Adapter DWMBA eingesetzt werden, die den von DEC vor vier Jahren eingeführten VAXBI-Bus (10 MByte/s) herausführen. Damit ist die ganze Palette von peripheren Geräten verfügbar, die für die VAX-Systeme der Serien 6000 und 8000 entwickelt wurden. Insgesamt sind bei 4 XMIs bis zu 14 VAXBI-Bussysteme möglich.
Weiterhin wurden neue Adapter bzw. Controller zum unmittelbaren Anschluß an den XMI vorgestellt, die es gestatten werden, die volle E/A-Transferrate auszuschöpfen.
Mit dem neuen KDM70-Speichercontroller können Hochleistungsmagnetplatten, optische Speichersysteme und eine Platte aus dynamischen Speicherschaltkreisen (Solid State Disk) bedient werden. Zur Erhöhung der Leistungsfähigkeit eines VAXclusters wurde ein neuer intelligenter Computer-Interconnect-Adapter CIXCD angekündigt, der auch die beiden redundant ausgelegten Leitungen des CI-Interfaces für die Datenübertragung nutzt. Damit wird die Netto-Transferrate gegenüber bisher eingesetzten Adaptern auf das Vierfache gesteigert. Auch auf dem Gebiet der Netzwerkkommunikation gibt es eine Innovation. Der neue Ethernet-Adapter DEC LANcontroller 400 für die Serie VAX 9000, der die Ethernet-Bandbreite von 10 MBit/s fast vollständig ausschöpfen kann, ist auf der Basis eines eigenen VAX-Chips in CMOS-Technik hergestellt. Der vorgestellte Vektorprozessor realisiert 63 Vektorbefehle, die eine Erweiterung des VAX-Befehlssatzes darstellen. Dadurch bleibt die bisher bewährte Kompatibilität innerhalb der gesamten VAX-Familie erhalten. Auf den Systemen ohne Vektorprozessor werden die Vektorbefehle vom Betriebssystem VAX/VMS emuliert. Damit können Anwendungen, die die Vektorbefehle nutzen, auf jedem beliebigen VAX-Rechner abgearbeitet werden. Ein weiterer Vorteil besteht auch darin, daß solche Anwendungen auf jeder VAX-Workstation entwickelt und getestet werden können.
Zur Unterstützung des Vektorprozessors ist eine neue Version des VAX-FORTRAN-Compilers angekündigt, der automatisch eine Vektorisierung innerhalb eines Programms vornimmt. Dieser Compiler kann Programme zur effektiven Abarbeitung in einem Multiprozessorsystem auch automatisch parallelisieren. Durch die Vektorisierung der Daten und Verwenden des Vektorprozessors kann je nach Anwendung eine 3- bis 40fache Geschwindigkeitssteigerung erreicht werden. Typische Anwendungen sind die Signalverarbeitung, Wettervorhersagen, seismische Analysen, Finite-Elemente-Analysen usw., aber auch ökonomische Anwendungen, in denen vielfach Vektoren eingesetzt werden. Die technologische Basis der neuen VAX-Systeme ist die ECL-Technik (Emitter Coupled Logic). Die Chips weisen eine Fläche von 2,6 cm^2 mit ca. 10 000 Gattern auf. Die Schaltzeit beträgt ca. 200 Picosekunden und die interne Taktrate 500 MHz.
Der grundlegende technologische Fortschritt wurde durch die Entwicklung einer Multi-Chip-Unit-Technologie (MCU) erreicht. Die Chips werden dabei auf einen besonderen Träger, den High Density Signal Carrier (HDSC), aus Polyamid mit Kupferleiterzügen aufgebracht. Die Leiterzüge haben eine Breite von ca. 0,03 mm. Der HDSC kann auf einer Fläche von ca. 32 cm^2 bis zu 72 Chips aufnehmen. Die HDSCs werden durch die MCU gekapselt. Damit sorgt die MCU für mechanische Stabilität und die für die ECL-Technologie kritische Wärmeableitung. Die quadratische MCU hat insgesamt 804 Pins.
Für die Realisierung der Rechnermoduln wird ein Leiterkartenformat von 60 × 60 cm^2 gewählt, auf dem bis zu 16 MCUs montiert werden können. Eine solche Leiterkarte bildet einen kompletten Prozessor der neuen VAX-9000-Serie inklusive des Vektor-Prozessors.
Durch die Weiterentwicklung der Multi-Chip-Packungstechnologie konnten die Signallaufzeiten zwischen den Chips entscheidend verkürzt und damit die hohen Prozessorleistungen erreicht werden. Laut Aussage von Dr. Robert M. Glorioso, Vizepräsident für Hochleistungssysteme der Mutterfirma in den USA, hat DEC damit eine Führungsrolle in der Welt übernommen. Die Fa. DEC habe im vergangenen Geschäftjahr mehr als 1,5 Mrd. Dollar, das heißt 12 % des Umsatzes, in Forschung und Entwicklung investiert.
Die Preise der neuen Systeme variieren zwischen 2,7 Mio DM für das Einstiegsmodell VAX 9000 Modell 210 und 12 Mio DM für das mit vier Skalar- und vier Vektorprozessoren ausgerüstete System VAX 9000 Modell 440. Das Modell 210 wird ab dem zweiten Quartal 1990 ausgeliefert. Die Modelle 410, 420 und 440 werden im dritten Quartal folgen. Vorerst wird nur das Betriebssystem VAX/VMS angeboten. Bis Jahresende 1990 soll für die neuen Systeme auch das Betriebssystem ULTRIX, das DEC-Betriebssystem auf der Basis der POSIX-Standards bzw. des UNIX-Industriestandards, angeboten werden.

Prof. Dr. Th. Horn

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
4. Jahrgang · ISSN 0232–2892

Mikroprozessortechnik, Heft 2 · 90

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR–1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Hans Weiß, Verantwortlicher Redakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2870203), Hans-Joachim Hill (Tel. 2870209); Sekretariat Tel. 2870381

Gestaltung Christina Bauer

Titel Tatjana Stephanowitz

Lizenz-Nr. 1710 des Presse- und Informationsdienstes der Regierung der DDR.

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

Redaktionsschluß 11. Dezember 1989

AN (EDV) 49837

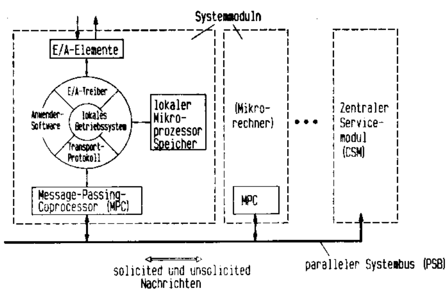
Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

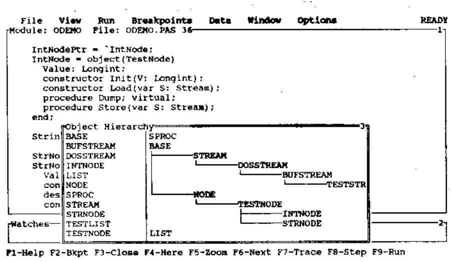
Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *Republik Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Republik Ungarn:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Hà Nôi; *BRD und Berlin (West): ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR–7010 Leipzig und Leipzig Book Service, Talstraße 29, DDR–7010 Leipzig

Zum Titelbild

In Magdeburg findet am 27. und 28. März 1990 die Fachtagung „Mikrorechner und ASICs in der Meß- und Automatisierungstechnik" statt. Es wird ein breites Spektrum aktueller technischer Resultate vorgestellt. Wir haben für Sie die auf einen breiten Leserkreis zugeschnittenen Fassungen von zwei Hauptvorträgen in dieses Heft aufgenommen.
Der Beitrag „Mikroelektronik und Meßwerterfassung" auf der **Seite 35** – auf den auch unser Titelbild Bezug nimmt – stellt die Haupttendenzen bei der Meßwerterfassung, insbesondere unter Einsatz von anwendungsspezifischen Schaltkreisen, dar.



Einen Vergleich von Bussystemen und ihre Perspektiven – mit dem Schwerpunkt auf den Bussystemen für Multiprozessoranordnungen – vermittelt der Beitrag auf der **Seite 39**.



Kurz nach der Markteinführung der Version 5.0 zeigt sich jetzt Turbo-Pascal 5.5 auch objektorientiert. Einige Möglichkeiten dieser erweiterten Programmiersprache werden in dem Beitrag zum neuen Compiler von Borland auf der **Seite 44** an praktischen Beispielen erläutert.

Vorschau

Für unser Heft 3/1990 bereiten wir für Sie Beiträge zu folgenden Themen vor:
- Clipper
- Systemprogrammierung in Modula-2
- Die serielle Schnittstelle unter DOS

Inhalt

MP-Dialog 34

Manfred Seifart:
Mikroelektronik und Meßwerterfassung 35

vorgestellt 38
Macintosh Portable

Wolfgang Meiling:
Bussysteme für Mikroprozessoranordnungen 39

Peter Lange, Dieter Büttner:
Die Dateihierarchie bei größeren Softwareprojekten unter Unix 42

vorgestellt 43
Microcontroller SAB 80C166

Uwe Schulze:
Turbo-Pascal 5.5 44

Wegbereiter der Informatik
Sir William Thomson (Lord Kelvin) 46

MP-Kurs: 47
Hartmut Pfüller, Wolfgang Drewelow, Berhard Lampe, Ralf Neuthe, Egmont Woitzel:
Einführung in Forth-83 (Teil 6)

Lutz Göhrs:
Kopplung von dBase III mit Supercalc 3 51

Thomas Streubel, Wolfgang Blume:
Komfortable Menüs für die Tabellenberechnung 52

Messeführer LFM '90 53

MP-Computer-Club 54
Eberhard Bauer:
Wordstar nutzt alle Drucksteuerzeichen
Hartmut Schröter:
Druckausgabe mit Turbo-Pascal umleiten
Bernd Matzke:
Bit-Funktionen unter Turbo-Pascal

Entwicklungen und Tendenzen 56

MP-Literatur 58

MP-Börse 60

MP-Bericht 61
32. Zentrale Messe der Meister von morgen
SYSTEMS 89
Erste europäische Amiga-Computermesse

Dialog

Liebe Leserinnen und Leser!

Unsere Medienlandschaft scheint von tektonischen Beben erschüttert. Vieles ist ins Wanken geraten, etliches bereits eingestürzt. Fast nichts ist heute, im Dezember 1989, noch so, wie es vor dem Oktober war. (Die Herstellzeit der MP ist allerdings noch die gleiche; deshalb wird wohl auch der Schnee, der bis jetzt noch gar nicht gefallen ist, schon wieder geschmolzen sein, wenn Sie dieses Heft in die Hände bekommen.) Die Kollegen des ADN haben mit ihren vehementen Forderungen nach Erlaubnis zu eigenverantwortlicher, ungeschönter, sachlicher Berichterstattung zur Wende beigetragen. Die Mitarbeiter der Aktuellen Kamera entschuldigten sich öffentlich für den Stil ihrer bisherigen Beiträge und zeigen jetzt, daß sie es auch anders können. Und was die Veränderungen in der Tagespresse angeht, so sprechen diese für sich.

Die MP-Redaktion: Hans Weiß, Verantwortlicher Redakteur, Herbert Hemke und Hans-Joachim Hill, Redakteure, und – last but not least – unsere neue redaktionelle Mitarbeiterin Hannelore Kulik

Hier drängt sich natürlich die Frage nach Konsequenzen für die Fachpresse und demnach auch für die MP auf. Würden wir als zuverlässiges Barometer für unsere Arbeit lediglich die Leserresonanz ansehen, müßten wir zufrieden sein: Die Auflage unserer Zeitschrift liegt seit jeher stabil bei weit über 50 000 Exemplaren, ein Beweis für Ihre Treue zur Zeitschrift, und der Anteil kritischer Leserpost ist so gering wie nie zuvor. Vielleicht ist dies ein Zeichen dafür, daß wir die Frage: Was nützt dem Leser wirklich? nicht erst nach der Wende stellen (können), sondern von Beginn an permanent stellten. Dadurch konnten viele nützliche Hinweise von Ihnen die Profilierung der MP bewirken. Erinnert sei nur an die nach Leserdiskussionen vorgenommene Reduzierung von Beiträgen mit vordergründiger Selbstdarstellung einer Leistung oder Produktwerbung. Oder an die Tatsache, daß wir bei der Annahme – und noch wichtiger: Ablehnung – von Beiträgen den Nutzen für einen bestimmten Hersteller nicht generell mit dem Nutzen für die Leser gleichsetzten, sondern versuchten, verantwortungsbewußt unter gesamtvolkswirtschaftlichen Gesichtspunkten abzuwägen. Demzufolge orientierten wir uns nicht an einer gewünschten oder vorgetäuschten, sondern an der materiell-technischen Basis, die unsere Leser in der Wirklichkeit vorfanden. Dabei wird es natürlich auch bleiben. Darüber hinaus glauben wir aber, daß unter den veränderten Bedingungen in unserem Lande durchaus weitere Verbesserungen notwendig und möglich sind.

Über diese Möglichkeiten und Pläne wollen wir Sie an dieser Stelle kurz informieren, so wie wir es in den jeweiligen Februarheften der letzten Jahre im Dialog mit Ihnen getan haben. Übrigens: Ist auch Ihnen bewußt, daß wir bereits auf drei Jahrgänge der MP zurückblicken können? Wir haben in der Redaktion dieses kleine Jubiläum zum Anlaß für eine Rückschau genommen, um auch daraus Schlußfolgerungen zu ziehen. Außerdem meinen wir, daß Sie nun doch einmal Gelegenheit erhalten sollten, die Mannschaft der MP kennenzulernen und liefern Ihnen daher auf dieser Seite unsere Konterfeis.

Was kann, was soll sich nun ändern? Beispielsweise die Rubrik Informationen. Hier waren und sind wir weitgehend auf Meldungen der Nachrichtenagentur ADN angewiesen, zu der ich eingangs schon etwas sagte. Dabei muß aber auch bedacht werden, daß der Wille zu objektiver Berichterstattung nur eine Seite der Medaille ist. Die andere zeigt sich in dem, was den Journalisten des ADN und der übrigen Presse in den Betrieben „vorgeführt" wird. So wird es hoffentlich in Zukunft möglich, aber nicht mehr nötig sein, Schönfärberei in Erfolgsmeldungen zu beseitigen. Andernfalls werden wir auf diese „Informationen" verzichten. In dieser Hinsicht setzen wir auch große Hoffnungen in das erwartete Mediengesetz.

Das Gesagte gilt in hohem Maße auch für die anderen Beiträge. Eine wissenschaftlich-technische Fachzeitschrift kann immer nur Mittler zwischen den Autoren und den Lesern sein, da die überwiegende Mehrzahl der Manuskripte von Fachleuten außerhalb der Redaktion verfaßt wird. Das zeigt, daß die Qualität der Fachzeitschrift vor allem von der Bereitschaft ihrer Leser zur Mitarbeit als Autoren geprägt wird, zumindest, so lange freie Fachjournalisten mit einem Gespür für leserfreundliche Aufbereitung von Fachwissen in unserem Lande (noch?) rar sind. Vielleicht können also auch Sie, der Sie Leser der MP sind, selbst als Autor zur Qualifizierung Ihrer Fachkollegen beitragen? Der Redaktion obliegt es, in Zusammenarbeit mit dem ehrenamtlichen Redaktionsbeirat, den Autoren und Lesern, die „richtigen" Manuskripte zu beschaffen und auszuwählen.

Im Heft 2/1989 hatten wir unsere Meinung zum Profil der MP ausführlich dargelegt. Dennoch gibt es bei einigen Lesern Vorstellungen, denen wir gegenwärtig leider nicht entsprechen können. So wird das Profil natürlich auch wesentlich von Faktoren wie Seitenzahl und Manuskriptangebot bestimmt. Und diese zwangen uns, zugunsten unserer Hauptlesergruppe – Hard- und Softwareentwickler – bereits seit Anfang 1988 Einschränkungen bei wissenschaftlich-theoretischen Beiträgen einerseits und bei der Kleincomputerthematik andererseits vorzunehmen. Deshalb will ich die Ausführungen zum Charakter der MP hier noch einmal kurz zusammenfassen:

- Die MP ist eine wissenschaftlich-technische Fachzeitschrift, die vor allem einen berufsmäßig vorgebildeten Leserkreis in seiner praktischen Tätigkeit unterstützt.
- Angesprochen werden alle ingenieurtechnischen Mitarbeiter, die auf dem Gebiet der Mikroelektronik, Computertechnik und Informatik tätig sind.
- Die MP informiert demzufolge über die spezifischen Bauelemente der Computertechnik und deren Applikationen, über Entwicklungen auf dem Computersektor, etwa zu gleichen Teilen Hardware und Software, sowie zukunftsorientierend zu Fragen der Informatik.

Erreichen wollen wir das mit einem ausgewogenen Verhältnis von Fachaufsätzen, Überblicksbeiträgen und informativen Rubriken. An Themen stehen dabei demnächst folgende auf der Tagesordnung:

- dBase-Compiler Clipper im Überblick
- das 32-Bit-Mikroprozessorsystem U 80700
- der 32-Bit-Rechner K 1820
- Mikrocontroller.

Darüber hinaus die Komplexe lokale Netze, Leiterplattenentwurf am PC, Applikationserfahrungen mit den Prozessorsystemen 80286 und U 80600, Signalprozessoren.

Es scheint uns im Sinne der Mehrzahl unserer Leser zu sein, wenn wir versuchen, die bewährten Rubriken wie Börse, Entwicklungen und Tendenzen, Messeberichte, Technik international usw. nicht einzuschränken, sondern nach Möglichkeit noch zu erweitern.
An dieser Stelle noch zwei kurze Bemerkungen zur Rubrik Börse. Zum einen wollen wir nochmals darauf hinweisen, daß wir im Interesse einer hohen Aktualität Angebote nur veröffentlichen, sofern sie nicht auch in einer anderen Zeitschrift erscheinen. Zum anderen machten uns Leser auf vermutlich unseriöse Softwareangebote aufmerksam und schlugen vor, vom Anbieter eine Erklärung zur Rechtsmängelfreiheit zu verlangen. Die Redaktion hält dies für einen kaum wirksamen zusätzlichen Aufwand; man sollte davon ausgehen, daß sich jeder, der unrechtmäßig Software zum Verkauf anbietet, über die Konsequenzen klar ist, die sich daraus für ihn ergeben können.

Auch Ihnen wird es sicher nicht entgangen sein: Unsere Presselandschaft wird gegenwärtig unübersehbar um eine neue Facette bereichert – Anzeigen aus dem westlichen Ausland. Auch wir haben über diese neue Möglichkeit nachgedacht, meinen aber, daß wir in Anbetracht des bescheidenen Umfanges der MP keine Seiten für Anzeigen opfern sollten. Vorbehaltlich allerdings, solange nicht geregelt ist, daß sich die Anzeigenerlöse zugunsten unserer Leser direkt oder zumindest erkennbar in einer Qualitätserhöhung der Zeitschrift niederschlagen. Aber auch hierbei werden wir unsere Entscheidung anhand des Leserechos ständig überprüfen und bitten Sie daher um Ihre Meinung.

Abschließend möchte ich Sie schon auf ein kleines Bonbon neugierig machen, das Sie ab Heft 4/1990 erwartet: Im Zuge wachsender Kooperationsmöglichkeiten haben wir als „Start" in diese Richtung mit der anerkannten BRD-Fachzeitschrift *c't – Magazin für Computertechnik* eine Vereinbarung getroffen, die es uns erlaubt, Beiträge aus der c't künftig für die MP zu übernehmen. Nicht weil es vielleicht Mode ist, sondern weil wir uns davon eine Bereicherung der MP versprechen, die für viele von Ihnen von Nutzen sein wird. Denn auch für den grenzüberschreitenden Austausch von Fachwissen sollte sich besonders eine Computerzeitschrift mehr denn je offenhalten, meint Ihr

Hans Weiß

Hans Weiß

Mikroelektronik und Meßwerterfassung

Prof. Dr. Manfred Seifart, Magdeburg

Die Leistungssteigerung integrierter Schaltkreise in Verbindung mit der intensiven Weiterentwicklung der Software bewirkt gegenwärtig und in den nächsten Jahren einen tiefgreifenden Wandel in der Meß- und Automatisierungstechnik. Für die Meßwerterfassung sind folgende Haupttendenzen erkennbar:
① weitere Beschleunigung des Übergangs zur dezentralen prozeßnahen Meßwerterfassung und Aufgabenverteilung in zu automatisierenden Prozessen; Hauptbestandteile dieser dezentralen Systeme sind intelligente Module zur Meßgrößenerfassung und zur Prozeßbeeinflussung (Prozeßkoppelmodule /1/, /2/, /3/), ein einfacher Feldbus /4/, /5/ und eine Zentrale (meist PC), die das System steuert und die Verarbeitung und die Ausgabe der Meßwerte übernimmt und ggf. die Kopplung zu anderen Systembestandteilen gewährleistet.
② Der Anteil von Geräten mit eigener „Intelligenz", die über serielle Busse gekoppelt und gesteuert werden können, wird stark zunehmen. Solche Einheiten sind nicht nur wesentlich leistungsfähiger und flexibler einsetzbar als frühere Gerätelösungen, sondern sie verfügen auch über immer leistungsfähigere und wirkungsvollere Eigenschaften wie Selbstkalibrierung, umfangreiche Datenvorverarbeitung, Fernsteuerbarkeit und Kommunikationsfähigkeit mit nahezu beliebigen Geräten und Rechnern (Vernetzung ganzer Systeme). Auch fehlertolerante Systeme und der Einsatz von Expertensystemen werden zunehmend eine größere Rolle spielen.
③ Der PC wird infolge seiner rasch wachsenden Leistungsfähigkeit, sinkender Preise und zunehmend leistungsfähigerer Software sowohl für den CAD-Entwurf von ASICs, Baugruppen und Systemen der Meßwerterfassung als auch zur direkten Lösung von Meß-, Steuer- und Regelaufgaben große Bedeutung erlangen. Es erfolgt eine Verdrängung der klassischen, speziellen Meßgeräte durch „virtuelle Instrumente".
Drei Konzeptionen des PC-Einsatzes in der Meßtechnik lassen sich unterscheiden:
- dezentrale Meßwerterfassung/Automatisierungssysteme mit räumlich verteilten (typisch 10–50) Prozeßkoppelmodulen, mit digitalen Meßumformern oder intelligenten Sensoren bzw. Sensoren mit serieller Schnittstelle, die über einen Feldbus oder einen Sensorbus (meist verdrillte Zweidrahtleitung) mit einer Zentrale kommunizieren, wobei die Entfernung zwischen den Prozeßkoppelmodulen untereinander und zur Zentrale mehrere 100 oder über 1000 Meter betragen kann
- sogenannte PC-Meßgeräte ohne eigene Bedien-, Anzeige- und Auswertefunktionen, bei denen ein leistungsfähiger PC – meist über die IEC-Bus-Schnittstelle gekoppelt – diese Funktionen für mehrere unterschiedliche PC-Meßgeräte (Meßteil der eigentlichen Meßanordnung) übernimmt und eine Darstellung „virtueller" Frontplatten auf dem PC-Bildschirm erfolgt (Tafel 1) /6/
- Einsteckleiterkarten mit leistungsfähiger Hard- und Software zur Prozeßkopplung bzw. Meßwerterfassung; sie haben international bereits eine beachtliche Leistungsfähigkeit erreicht. Beispiel: Datenerfassungs- und Datenanalyseleiterkarte der Firma Datalog (DAP-Serie); unter Einsatz des 16-Bit-Prozessors 80186 werden 150 000 Messungen/s mit 16-Bit-Auflösung erreicht; die Leiterkarte beinhaltet einen 512-KByte-Datenspeicher und ein Echtzeitbetriebssystem sowie mehrere vordefinierte Meßwertverarbeitungsprogramme im ROM. Die PC-XT/AT-kompatible Leiterkarte ermöglicht eine hundertfach schnellere Datenverarbeitung als sie mit dem PC allein möglich wäre.
④ verstärkter Einsatz von ASICs vor allem für Steuerfunktionen in der Meßwerterfassung und als Sensor-Interfaceelektronik /7/.
Nachfolgend wird der Trend zur dezentralen prozeßnahen Meßwerterfassung und zum Einsatz von ASICs an vier Beispielen aus den Forschungsarbeiten des Wissenschaftsbereiches Prozeßmeßtechnik der Technischen Universität Magdeburg verdeutlicht.

Tafel 1 Aufgabenverteilung bei PC-Meßgeräten

PC-Meßgeräte	
• Meßwerterfassung	• AD-Umsetzung
• Statusinformation	• Stimulierung des Prüflings
Personalcomputer	
• Parametrierung (Meßgeräte, Meßplatze) • Abspeichern von Parametersätzen (Diskette, Festplatte) • Steuerung der Messungen, Triggerung • Meßgerätebedienung; grafische Darstellung der Bedien- und Anzeigeelemente auf Bildschirm • Abspeichern von Meßergebnissen in Dateien, Ausgabe auf PC-Peripherie • Signalverarbeitung, komplexe Weiterverarbeitung, Auswertung der Meßergebnisse • Formatieren von Datensätzen, Weiterverarbeitung mit Standardprogrammen der Meßwertverarbeitung	

Programmierbarer Prozeßkoppelmodul PPM 02

Die unter anderem im Energiekombinat Rostock erfolgreich erprobten und im Rahmen eines Meßwerterfassungs- und Automatisierungssystems eingesetzten Prozeßkoppelmodule mit Einchipmikrorechner PPM 01 /1/ lassen sich sowohl für Meß- als auch für Steuer- und Zählaufgaben, zur Grenzwertüberwachung, Regelung u. a. m. verwenden. Sie verfügen über einen galvanisch getrennten Feldbusanschluß, über den die Kommunikation in beiden Richtungen erfolgt. Bis zu 128 PPMs lassen sich räumlich an beliebiger Stelle an eine einfache, verdrillte Zweidrahtleitung anschließen. Die Systemsteuerung übernimmt ein PC.
Nachdem sich der Typ PPM 01 im industriellen Einsatz gut bewährt hat und inzwischen in die Serienproduktion (Energiekombinat Gera) übergeleitet wurde, ist gegenwärtig die Überleitung der weiterentwickelten Variante PPM 02 (Bild 1) im Energiekombinat Rostock im Gange. Der PPM 02 zeichnet sich vor allem durch Sensorsignalverarbeitung, erweiterte Speicherkapazität und ein verbessertes Softwarekonzept aus. Er verfügt über 6 Sen-

Am 25. März diesen Jahres feiert der Autor dieses Beitrages, Prof. Dr. sc. techn. Manfred Seifart, seinen 60. Geburtstag.
In der Zeit von 1951 bis 1956 studierte er an der Technischen Universität Dresden Hochfrequenztechnik. Danach arbeitete er 13 Jahre mit hohem Engagement im Wissenschaftlichen Industriebetrieb VEB Vakutronik Dresden (heute VEB Robotron Meßelektronik „Otto Schön") auf dem Gebiet der Entwicklung kernphysikalischer und elektronischer Meßgeräte. Von seiner Funktion als Entwicklungsleiter erfolgte im Jahre 1969 die Berufung zum Ordentlichen Professor an die damalige Technische Hochschule Magdeburg auf den Lehrstuhl „Elektronische Bauelemente und Schaltungstechnik". Seit 1984 ist er Leiter des Wissenschaftsbereiches Prozeßmeßtechnik in der Sektion Automatisierungstechnik und Elektrotechnik.
Die von Prof. Seifart an der Technischen Universität Magdeburg eigenständig begründete Forschungslinie beinhaltet den Schaltungsentwurf von Geräten zur industriellen Meßwerterfassung mittels Mikrorechnern. Dabei erzielte er bemerkenswerte, national und international anerkannte Ergebnisse bei der Entwicklung von neuartigen Analog-Digital- und Digital-Analog-Umsetzern, von Analogwerterfassungseinheiten für Mikrorechner sowie von mikrorechnergestützten Meßwerterfassungssystemen für die Prozeßmeßtechnik.
Besonders engagiert sich Prof. Seifart für den Entwurf von Schaltkreisen in der Gate-Array-Technik. Die Arbeiten zur Anwendung kundenspezifischer Schaltkreise beeinflußten stark das Profil der Sektion und haben wesentlichen Einfluß auf den Einsatz der Mikroelektronik im Schwermaschinen- und Anlagenbau. Seine wissenschaftliche Ausstrahlungskraft hat er durch über 150 Fachartikel und Vorträge im In- und Ausland unter Beweis gestellt. Mehr als 20 Patente unterstreichen die volkswirtschaftliche Bedeutung seiner Forschungsergebnisse. Die von Prof. Seifart überzeugend vertretene Lehrauffassung einer betont anwendungsorientierten Ausbildung der Studenten hat ihren Niederschlag unter anderem in seinen beiden umfangreichen Lehrbüchern über analoge und digitale Schaltungstechnik gefunden, die im VEB Verlag Technik und in mehreren Lizenzausgaben des Hüthig-Verlages erschienen sind.
Prof. Seifart gehört seit über 20 Jahren zu den hervorragenden Hochschullehrerpersönlichkeiten der Technischen Universität „Otto von Guericke" Magdeburg. So bewährte er sich 7 Jahre als Stellvertreter des Sektionsdirektors für Erziehung, Aus- und Weiterbildung und leistete dabei eine verantwortungsbewußte erzieherische Arbeit. Prof. Seifart ist Vorstandsmitglied der wissenschaftlichen Sektion „Computertechnik" der KDT und hat als Leiter des Fachausschusses „Mikroprozessor-Interface-Systeme" wesentlich zum breiten Erfahrungsaustausch auf diesem Gebiet beigetragen. Als Mitglied des Redaktionsbeirates der Fachzeitschrift „Mikroprozessortechnik" leistete er bei ihrer Profilierung einen großen Beitrag.
Als Wissenschaftler und Hochschullehrer gilt sein besonderes Interesse der Herausbildung des wissenschaftlichen Nachwuchses. Viele ehemalige Doktoranden sind heute in verantwortungsvollen Funktionen in Einrichtungen unseres Landes tätig.
Prof. Seifart, der mit viel Weitsicht und Umsicht die Entwicklung seines Fachgebiets maßgeblich gefördert hat, ist weit über die Grenzen der Technischen Universität Magdeburg hinaus anerkannt, geschätzt und geachtet. In seiner Aufrichtigkeit und Liebenswürdigkeit ist er vielen ein Beispiel.

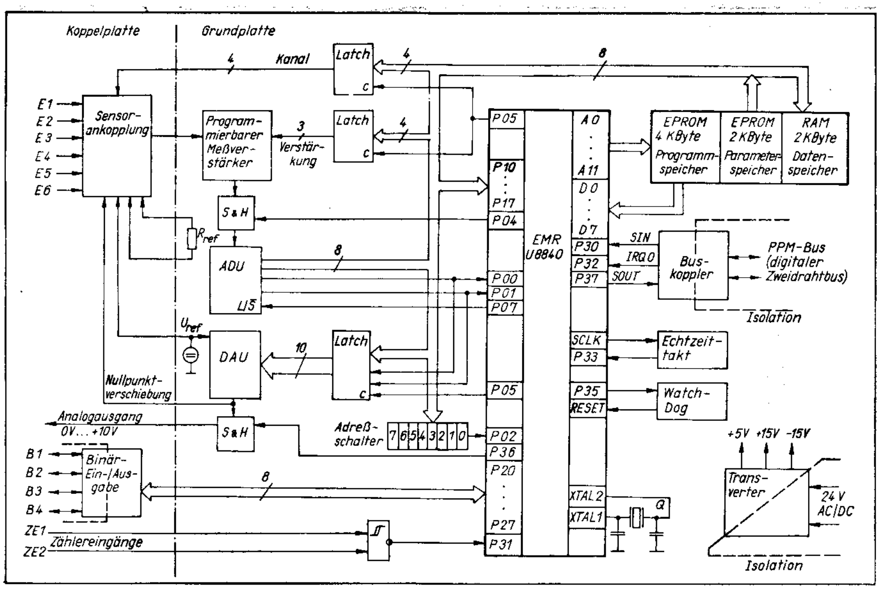


Bild 1 **Blockschaltbild Prozeßkoppelmodul PPM 02**

soreingangskanäle, 2 Zähleingänge, 4 (8) binäre Ein-/Ausgänge, 1 erweiterbaren Analogausgang sowie den galvanisch getrennten Feldbusanschluß und die galvanisch getrennte Betriebsspannungszuführung (24 V≂).

Die Sensorsignale werden in einem programmierbaren Meßverstärker, der in eine automatische Meßkette einbezogen ist, verstärkt. Mit einem 10-Bit-DAU (Digital-Analog-Umsetzer) ist der Meßbereichsanfang verschiebbar, so daß ein gewünschter Meßbereichsausschnitt mit der vollen Auflösung des 10-Bit-ADUs (Analog-Digital-Umsetzer) digital weiterverarbeitet werden kann. Zusätzlich erfolgt ein automatischer Offsetabgleich. Eine am Eingang verfügbare Stromquelle wird zur Speisung passiver Sensoren mit in das Abgleichregime einbezogen. Das beim PPM 01 bewährte Konzept der konstruktiven Unterteilung in eine Grund- und eine Koppelplatte wurde beibehalten.

Die räumliche Nachbarschaft zwischen den Sensoren bzw. Aktoren und dem Einchipmikrorechner hat zahlreiche Vorteile: Sie ermöglicht unter anderem die automatische Nullpunkt- und Vollausschlagkorrektur, die Überwachung der Eingangsschaltung und gegebenenfalls die Fehlerkorrektur sowie eine sehr zuverlässige, hochgenaue Signalübertragung über große Entfernungen. Weiterhin gestattet sie mit weitgehend einheitlicher Hardware eine Vielzahl unterschiedlicher Konfigurationen ausschließlich durch Softwareanpassung.

Softwarekonzept

Der Trend zur dezentralen Meßwerterfassung hat zur Folge, daß der Aufwand für die Softwareentwicklung stark ansteigt. Der PPM-Anwender möchte verständlicherweise vermeiden, sich in das detaillierte Hard- und Softwarekonzept des PPM einarbeiten zu müssen. Unter diesen Gesichtspunkten wurden Überlegungen zur Softwaregestaltung des PPM 02 angestellt. Es zeigte sich, daß die Signalverarbeitung zwischen dem Sensor/Aktuator und dem Applikationsprogramm des Anwenders in Analogie zum OSI-Schichtenmodell der Kommunikationstechnik in 8 Schichten mit klar definierten Interfaces zwischen den verschiedenen Softwarekomponenten unterteilt werden kann /8/ (Bild 2). Als Ergebnis einer Analyse der Informationsflüsse und Operationen wurden drei Kanaltypen definiert: Die globalen Informationskanäle sind nur für den Programmierer von zentralen Applikationsprogrammen interessant. Die lokalen Informationskanäle dienen dem Programmierer dezentraler Programme, und schließlich sind die Prozeßkanäle nur für den Programmierer von Firmware des PPMs von Interesse. Jedem Kanaltyp sind typische Softwarekomponenten zugeordnet. Am unteren Ende des Schichtenmodells befinden sich die reellen Prozeßkanäle. Die darüberliegenden Schichten lassen sich wie folgt charakterisieren:

Schicht 8

Applikationsprogramme

Schicht 7

Protokollsoftware Master/PPM (teilweise); Umwandlung lokaler in globale Kanäle

Schicht 6

Mailbox repräsentiert das Interface zum seriellen Bus; alle für das zentrale Anwenderprogramm benötigten Informationen sind verfügbar; Master greift nur auf die Mailbox zu.

Schicht 5

Dezentrale Anwenderprogramme (Kombinieren mehrerer Prozeßkanäle; Softwarefilterung, Grenzwertüberwachung; Einschreiben in die Mailbox u. a.)

Schicht 4

Einrichten der umgesetzten Signale in der Anwendungsschicht (Skalieren, Normieren, Offsetberechnung, Linearisieren u. a.)

Schicht 3

Umsetzen (ADU, DAU, Zählen, Zeitintervallbestimmung, ADU-Steuerung u. a.)

Schicht 2

Anpassen der Sensorsignale an den ADU-Eingang (Verstärken, Begrenzen, Meßbereichsgrenzen, I → U- bzw. R → U-Wandlung)

Schicht 1

Anschließen der Sensoren an die Koppelleiterkarte des PPM (Verkabelung, Torauswahl)

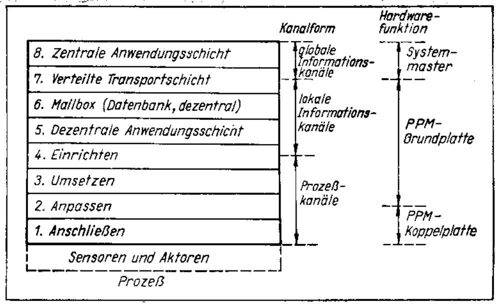


Bild 2 **Schichtenmodell der dezentralen Meßdatenverarbeitung mit PPM**

Gate-Array mit seriellem Busanschluß für die Meßwerterfassung

Zur Ankopplung von Sensoren an einen Feldbus ist ein erheblicher Schaltungsaufwand nötig, so daß der Einsatz anwendungsspezifischer Schaltkreise sehr erwünscht ist. Allerdings werden neben umfangreichen digitalen Schaltungen auch hochpräzise Analogschaltungen benötigt, so daß deren komplette Realisierung in einem CMOS-Gate-Array gegenwärtig noch mit einer Reihe von Problemen behaftet ist. Das im Bild 3 gezeigte Gate-Array verdeutlicht, daß es durchaus möglich ist, mit einem ausschließlich für digitale Schaltungen konzipierten Gate-Array-System relativ niedriger Komplexität (U 5200) einen Sensor-Interfaceschaltkreis mit seriellem Busanschluß herzustellen, der einen 12-Bit-ADU (Digitalteil) beinhaltet, eine 2-Byte-CRC-Datensicherung ermöglicht, in drei unterschiedlichen Betriebsarten (stand alone, non stop, Einchipmikrorechnerkopplung) einsetzbar ist und mit einer einzigen Betriebsspannung von 5 V auskommt.

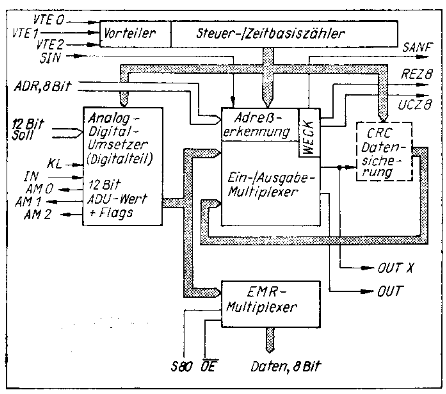


Bild 3 **Blockschaltbild des Meßwerterfassungs-Gate-Array-Schaltkreises**

Grobstruktur

Der Meßwerterfassungsschaltkreis besteht aus den drei Schaltungsgruppen Steuerzähler mit 5-Bit-Vorteiler, ADU und Interfaceschaltungen (Bild 3). Um die relativ geringe Anzahl verfügbarer Flipflops maximal auszunutzen, gestattet der 14-Bit-Steuerzähler zwei weitgehend unabhängige Funktionen. Er bildet den 12-Bit-Zeitbasiszähler für den integrierenden ADU und generiert die erforderlichen Steuersignale für den synchronen Ablauf aller wesentlichen Schaltungsfunktionen. Diese Doppelfunktion erfordert eine zeitliche Synchronisation des gesendeten Busprotokolls mit den drei aufeinanderfolgenden ADU-Zyklen Offset-, Meßwert- und Endwertzyklus (je eine getrennte 12-Bit-Umsetzung).

ADU

Durch eine neuartige Schaltungsvariante beim verwendeten Charge-balancing-ADU konnte der Anteil analoger Schaltungsstrukturen extrem reduziert werden, so daß im einfachsten Fall nur 3 bis 4 externe Bauelemente angeschlossen werden müssen. Der Eingangssignalbereich beträgt $U_e = 0 \ldots 5$ $(\pm 2{,}5)$ V bzw. $I_e = 0 \ldots 100$ (± 50) μA.
Der ADU stellt die komplexeste Teilschaltung des Schaltkreises dar (50 % der Chipfläche). Er beruht auf dem Ladungsausgleichsverfahren /9/. Das hat unter anderem den großen Vorteil, daß dieses Verfahren mit einem hohen Anteil digitaler Schaltungsgruppen aufgebaut werden kann und sich deshalb gut für hohe Integrationsstufen wie VLSI eignet. Als Ergebniszähler wird ein 12-Bit-Vor-/Rückwärtszähler mit komplexer Ansteuerelektronik verwendet. Die Ansteuerelektronik erfüllt folgende Aufgaben: Zykluskodierung, Übertragsauswertung und Zählrichtungsumschaltung, Sollwert- und Kennlinieneinstellung, Parallelladesteuerung für den Ergebniszähler, Registerladesteuerung, Zeitbasisauswertung sowie Flagauswertung. Diese relativ komplexe Ansteuerung ist erforderlich, da die Möglichkeit gegeben ist, das analoge Eingangssignal wahlweise an einer bipolaren oder an einer unipolaren Kennlinie umzusetzen und periodisch ein Offset- und ein Endwertabgleich erfolgt. Das in der Offsetphase ermittelte Zählergebnis wird zu Beginn des Meßwertzyklus als Voreinstellwert in den Ergebniszähler geladen und durch Umschalten seiner Zählrichtung vom Endergebnis subtrahiert. Der Endwert läßt sich beim hier verwendeten Umsetzverfahren besonders einfach durch Voreinstellung des Teilerverhältnisses abgleichen. Als externe Bauelemente werden für den ADU im einfachsten Fall lediglich ein Integrationskondensator und zwei Präzisionswiderstände sowie bei höheren Genauigkeitsanforderungen zusätzlich ein Referenzelement und ein Multiplexer benötigt.

Busprotokoll

Zur Kommunikation auf dem Feldbus wurden die Bustopologie, ein zentrales Zugriffsverfahren (Polling) und die Basisbandübertragung im NRZ-Kode gewählt. Jedem Meßwerterfassungsschaltkreis wird vom Anwender eine 8-Bit-Adresse zugeordnet, die mit DIL-Schaltern eingestellt werden kann und die der Schaltkreis bei der Buskommunikation selbst erkennt. Die Datensicherung erfolgt durch ein 2-Byte-CRC-Verfahren mit dem Nennerpolynom $x^{16} + x^{12} + x^5 + x$. Das in der Betriebsart *stand alone* gesendete Busprotokoll zeigt Bild 4.

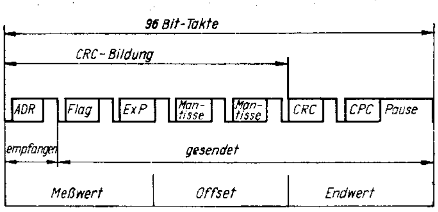


Bild 4 Busprotokoll in der Hauptbetriebsart „Stand alone"

Anwendungsbeispiele

Bild 5 zeigt die einfachste Möglichkeit des Anschlusses mehrerer Schaltkreise an einen seriellen Bus (Hauptbetriebsart stand alone). Der Eingang SIN liegt ständig am Feldbus. Das nach einer Pause von ≧ 15 H-Pegeln auf dem Bus eintreffende erste L-Bit wird als Startbit des Adreßbytes gewertet, und es erfolgt ein Adressenvergleich. Im Hintergrund laufen ständig zeitlich nacheinander die drei oben genannten ADU-Zyklen ab. Unmittelbar nach erkannter Startflanke wird der gerade in Arbeit befindliche Zyklus unterbrochen, und es beginnt sofort die Umsetzung des aktuellen Meßwertes. Falls der Schaltkreis adressiert wurde, wird ein Busprotokoll nach Bild 4 gesendet. In der Regel werden die Gate-Arrays galvanisch getrennt (mit Optokoppler oder Übertrager) an den Bus angekoppelt. Wegen der geringen Leistungsaufnahme des Arrays besteht die Möglichkeit, mit einem Übertrager sowohl die Information als auch die Hilfsenergie zu übertragen.

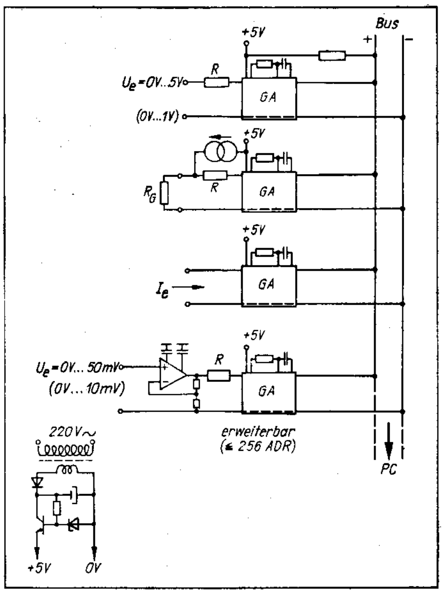


Bild 5 Einfachste Zusammenschaltung mehrerer Meßwerterfassungs-Gate-Arrays

Mehrkanalige programmierbare Meßwerterfassungs- und Prozeßkoppelmodule lassen sich konfigurieren, indem der Meßwerterfassungsschaltkreis zusammen mit einem Einchipmikrorechner betrieben wird. Der Schaltkreis übernimmt neben der AD-Umsetzung die Pausen- und Adreßerkennung auf dem Bus. Die ADU-Daten und 4 Flagbits werden byteweise einem Eingangstor des Einchipmikrorechners (U 884) zugeführt. Zusätzlich erzeugt der Schaltkreis bei gültiger Adresse zwei Wecksignale, die verwendet werden können, um den Einchipmikrorechner vom Standby- in den aktiven Betrieb umzuschalten. Dadurch besteht die Möglichkeit, in einem Meßwerterfassungssystem gleichzeitig nur wenige oder nur einen PPM zu aktivieren und alle übrigen in den Schlafzustand mit stark verringerter Leistungsaufnahme zu versetzen. Das bietet die Möglichkeit der gleichzeitigen Informations- und Hilfsenergieübertragung über den Bus.

Feldbuskoppler-Schaltkreis für Profibus

Große Bedeutung hat der Einsatz hochintegrierter Schaltkreise zur Schaffung einer standardisierten bzw. einheitlichen Feldbusschnittstelle. Einen ersten Schritt stellt der gegenwärtig in der Musterpräparation befindliche Feldbuskoppler-Gate-Array-Schaltkreis (U 5200) dar, der die beiden Schichten 1 und 2 des Profibus-Protokolls unterstützt. Er enthält die vier Hauptblöcke UART, Controllerblock, Speicher-Interface und Betriebsmodussteuerung. Der Schaltkreis übernimmt das Senden und das Empfangen des Protokolls, wobei eine Adressen- und Datenübertragungsfehlerbehandlung erfolgt. Die parallele Datenschnittstelle zum angeschlossenen Prozessor bildet ein RAM U 6516, der durch Steuerschaltungen, die sich im Gate-Array befinden, dualportfähig ist. Bild 6 erläutert die Funktionsabdeckung des Schaltkreises im Gesamtsystem.

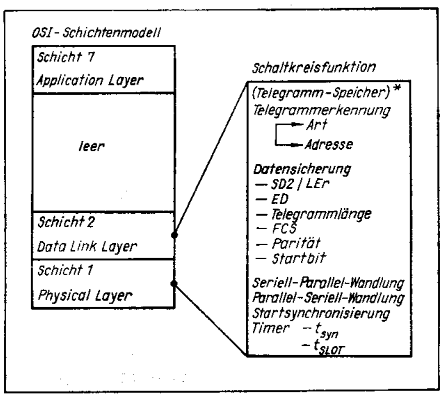


Bild 6 Funktionsabdeckung des Buskoppler-Schaltkreises im Gesamtsystem

Hochauflösende ADUs

Die seit Jahrzehnten verbreitete „klassische" Architektur von Mehrkanal-Analogwerterfassungssystemen mit typisch 8 bzw. 16 Eingangskanälen verwendet einen Analogmultiplexer im Eingangskreis, der jeweils einen Eingangskanal über einen programmierbaren Instrumentationsverstärker an den ADU-Eingang (typisch 12 Bit) schaltet. Die Leistungssteigerung integrierter Schaltkreise wird in den nächsten Jahren die Schaffung neuer Strukturen mit neuen Qualitäten ermöglichen, beispielsweise den Einsatz hochauflösender 16-Bit-ADUs in jedem Meßkanal, wodurch die galvanische Trennung jedes einzelnen Kanals bei vertretbarem Aufwand ermöglicht wird und darüber hinaus Analogmultiplexer, Instrumentations- bzw. Trennverstärker entfallen oder stark vereinfacht werden können. Verständlicherweise stellt die Entwicklung hochauflösender ADUs, die zusammen mit komplexen, leistungsfähigen Digitalschaltungen auf einem VLSI-Chip integriert werden, eine große Herausforderung für den Schaltungs- und Systementwickler dar. Erste Ergebnisse auf diesem Gebiet zeigen, daß Kombinationsverfahren, die auf dem Prinzip integrierender Umsetzer basieren, dieses Ziel in naher Zukunft erreichbar erscheinen lassen (Charge-balancing-Dual-Slope-Kombinationsverfahren / Mehrschritt-Charge-balancing-Verfahren / Deltasigma-Oversampling-Verfahren).

Trends bei der Software

Durch das starke Eindringen des PCs in meßtechnische Anwendungen werden Vielfalt und Komplexität zu lösender Meßaufgaben, die für die Meßwerterfassung typisch sind, dazu führen, daß der Umfang an benötigter

Software stark zunehmen wird. Zur Zeit existiert ein Rückstand gegenüber der Hardware. Der Meßtechniker wünscht sich verständlicherweise anwenderfertige Lösungen ohne Aufwand an speziellen Programmierkenntnissen. In den nächsten Jahren sind hier erhebliche Fortschritte zu erwarten. Gegenwärtig existieren als Bedienoberflächen für die Meßtechnik die drei Konzepte Menütechnik, Kommandosprachen und Windowtechnik, wobei letztere besonders leistungsfähig und bedienerfreundlich ist. Der Trend der nächsten Jahre wird markiert durch die in jüngster Zeit entwickelte Software für PC-Meßgeräte. Beispielsweise verwendet Siemens ein Grafiksystem mit GEM Desktop, das eine schnelle komfortable Bedienung mehrerer PC-Meßgeräte von einem PC aus ermöglicht. Teile von (gedachten) Meßgerätefrontplatten bzw. Anzeigen werden in Dateien hinterlegt und können vom PC auf Drukker ausgegeben werden. Meßkurven werden in eine GEM-Grafikdatei übernommen und können vom PC beliebig auf seine Standardperipheriegeräte oder auf den Bildschirm ausgegeben werden.

Auf dem Gebiet meßtechnischer Programmiersprachen ist ein Generationswechsel zu erwarten. Zunehmend werden grafische, objektorientierte Programmierwerkzeuge zur Entwicklung von Meßprogrammen entwikkelt. Ein Beispiel dafür ist der Programmgenerator PCI-Genesys (Siemens), der das automatische Generieren von PCI-Basis-Programmen ermöglicht, ohne daß der Anwender die Programmiersprache erlernen muß /6/. Der Anwender muß lediglich die Ablaufstruktur (logischer Programmablauf) und die Randbedingungen seiner Messungen grafisch am Bildschirm eingeben. Nach der grafischen Darstellung des gewünschten Meßablaufs wird automatisch ein PCI-Basic-Programm generiert. PCI-Basic ist eine speziell für IEC-Bus-gesteuerte Meßgeräte entwikkelte Programmiersprache mit einem Basic-Compiler zur freien Programmierung von PC-Meßplätzen und einzelnen PC-Meßgeräten.

Die hier erwähnte automatische Programmgenerierung und der breite Einsatz „virtueller" Instrumente als Software-Interfaces, das heißt die Darstellung mehrerer Frontplatten auf dem PC-Bildschirm, über die die angeschlossenen PC-Meßgeräte auch bedient und gesteuert werden können, werden in den 90er Jahren die Meßtechnik tiefgreifend beeinflussen.

Literatur

/1/ Seifart, M.: Intelligenter Prozeßkoppelmodul. Mikroprozessortechnik, Berlin 1 (1987) 1, S. 11

/2/ Beikirch, H. u. a.: Intelligente dezentrale Prozeßkoppelmodule. Feingerätetechnik, Berlin 26 (1987) 8, S. 360

/3/ Michaelis, B.; Maaß, R.: Meßwerterfassung mit CCD-Sensoren. Mikroprozessortechnik, Berlin 1 (1987) 2, S. 41

/4/ Güttler, F.: Digitale Meßwerterfassung und Kommunikation in der Kleinautomatisierung. 33. Internationale wissenschaftliche Konferenz der Technischen Hochschule Ilmenau 1988, Reihe A 2

/5/ Scholz, B.: Entwurf und Realisierung eines Feldbussystems für die Kleinautomatisierung. Technische Universität Magdeburg: Dissertation A, 1988

/6/ Simmel, H. E.: Programmerstellung mit dem Programmgenerator PCI-Genesys. 2. Kongreßmesse für industrielle Meßtechnik MessComp, 27.–29. 9. 1988, Wiesbaden, Tagungsband Vortrag 3A.5.

/7/ Seifart, M. u. a.: Einsatz anwendungsspezifischer Schaltkreise in Funktionseinheiten der Meßwerterfassung. Wissenschaftlich-technische Mitteilung VEB FER, H. 3 (1988) 3, S. 37

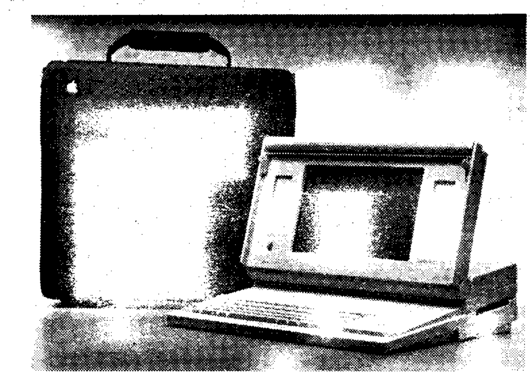
/8/ Fincke, U.: Funktionsprinzipien und -einheiten der dezentralen Meßwerterfassung. Dissertation A, Technische Universität Magdeburg 1988.

/9/ Seifart, M.: Analoge Schaltungen. 3. Aufl. Berlin: VEB Verlag Technik 1989.

⊠ KONTAKT ☏

Technische Universität „Otto von Guericke" Magdeburg, Sektion Automatisierungstechnik und Elektrotechnik, Wissenschaftsbereich Prozeßmeßtechnik, PSF 124, Magdeburg 3010; Tel. 59 21 31 43

vorgestellt **Macintosh Portable**



Apples erster Laptop

Die Anhänger von Computern der Firma Apple mußten lange warten, bis der erste Laptop vorgestellt wurde. Der Grund dafür, daß Apple als einer der letzten namhaften Hersteller nun erst ein Produkt dieser Kategorie herausbrachte, soll darin liegen, daß Apple auch hier Maßstäbe setzen wollte und dabei vor allem Probleme mit der Stromversorgung und dem Bildschirm gelöst werden mußten. Allseits anerkannt wird, daß dies gelungen ist. Der Portable präsentiert sich zum einen mit einem neuartigen 10-Zoll-Display – Active Matrix Liquid Crystal Display (AMLCD) genannt –, das sich bei einer Auflösung von 640 × 400 Pixeln vor allem durch den hohen Kontrast von 50 : 1, einen geringen Stromverbrauch und einen hohen Betrachtungswinkel auszeichnet. Dadurch, daß jeder Bildpunkt von einem eigenen, im Bildschirm integrierten Transistor angesteuert wird, also nicht impulsweise, besitzt das AMLCD nicht die von herkömmlichen LCDs bekannte Trägheit. Wie die PC-Woche berichtete, fertigen nur die beiden japanischen Firmen Hoseiden und Kyocera aktive Matrixdisplays. Die Tafel zeigt die bei Laptops zur Zeit eingesetzten Displaytechniken im Vergleich (Quelle: Apple). Zum anderen kennzeichnet den Mac Portable die außergewöhnlich lange netzunabhängige Betriebsdauer von bis zu 12 Stunden. Erreicht wird das unter anderem durch folgende Maßnahmen:

– Als CPU wird die stromsparende CMOS-Version Motorola 68HC000 verwendet (mit 16 MHz).

– Für den Hauptspeicher werden nicht die üblichen dynamischen Bausteine (DRAMs) eingesetzt, sondern statische CMOS-RAMs, die nicht ständig aufgefrischt werden müssen.

– Verwendet wird ein Bleiakku, der jederzeit auf volle Leistung nachgeladen werden kann. Dadurch, daß die Spannung bei diesem Batterietyp gleichmäßig abfällt, kann der Nutzer rechtzeitig automatisch vom kritischen Ladezustand unterrichtet werden (Maclike natürlich auf dem Monitor). Nach der vierten Alarmstufe begibt sich der Rechner in den Schlafmodus und gibt dem Nutzer fünf Tage Zeit, den Akku nachzuladen. Erst dann geht der Inhalt des RAMs verloren. (Übrigens kann der Akkublock in diesem Schlafmodus gewechselt werden, da eine 9-V-Batterie seine Aufgabe so lange übernimmt.)

– Für die Steuerung und Überwachung der Stromzufuhr setzt Apple eigens einen Mikroprozessor, den 50753, ein. Nach einem bestimmten Intervall, das vom Benutzer selbst festgelegt werden kann, schaltet der Prozessor das Gerät automatisch in den Sparbetrieb oder Ruhezustand. Der Prozessor läuft dann mit nur 1 MHz weiter. Die meisten anderen Komponenten werden ausgeschaltet – einschließlich der (optionalen) Festplatte mit einer Wiederanlaufzeit von weniger als drei Sekunden –, bis der Druck auf eine beliebige Taste den Rechner reaktiviert. Dieser Zeitraum kann mehrere Tage betragen.

Schon mit diesen Leistungen in bezug auf Nutzerfreundlichkeit übertrifft der Mac Portable viele seiner Konkurrenten. „Unser oberstes Ziel war, daß dieses Gerät seinen Namen verdient – Macintosh Portable. Das bedeutet, keinerlei Kompromisse zuzulassen," sagte Jean-Louis Gassée, Präsident der Apple Products Division bei der Vorstellung. Dafür sprechen auch die weiteren Merkmale: Die Tastatur ist mit der des bekannten Macintosh identisch; anstelle des Ziffernblocks wird jedoch standardmäßig eine (gegen den Ziffernblock austauschbare) Maus (Trackball) integriert. Neben dem FDHD-Diskettenlaufwerk (3,5 Zoll, 720 KByte oder 1,44 MByte), das alle gängigen Diskettenformate – auch MS-DOS – verarbeitet, kann ein zweites Disk- oder ein Festplattenlaufwerk (bis 40 MByte, 28 ms Zugriffszeit) installiert werden. Sieben Macintosh-Standardanschlüsse ermöglichen den Betrieb von Festplatte, Drucker oder Scanner. Darüber hinaus gibt es serielle Anschlüsse für einen Drucker, ein Modem oder das AppleTalk-Netz. Ein Modem-Slot oder Slots für RAM und ROM lassen Erweiterungen zu; allerdings nur mit einem Steckkartenformat, das nicht mit dem des bisherigen Macs oder anderer Hersteller übereinstimmt. Standardmäßig ist der Portable mit 1 MByte RAM ausgestattet; eine Aufrüstung ist derzeit auf 2 MByte möglich, nach der Verfügbarkeit von HD-Speicherchips auf 9 MByte. Die Maße des Laptops betragen 39 cm in der Breite, 38 cm in der Tiefe und 10 cm in der Höhe. Das Gewicht liegt (einschließlich Batterie und Festplatte) bei noch „schoßgerechten" 7,16 kg. Daß mit der Systemsoftware 6.04 volle Kompatibilität zum Betriebssystem, zur Benutzeroberfläche und zu den bekannten Anwenderprogrammen der übrigen Macs besteht, darf man bei Apple schon fast voraussetzen. Der Preis für all den Komfort liegt jedoch bei bis zu 16 000 DM. *MP-We*

Technologie	Leistungsaufnahme [W]	Kontrast	Blickwinkel [°]	Ansprechzeit [ms]
Gas-Plasma	20	20 : 1	45	16
Elektrolumineszenz	15	10 : 1	140	1
ST-LCD	5	20 : 1	25	200
Aktivmatrix	0,45	50 : 1	90	50

Bussysteme für Multiprozessoranordnungen

Prof. Dr. Wolfgang Meiling, Dresden

Bussysteme und Busstandards

Ein Bus stellt eine parallele Sammelleitung zur Übertragung von Daten, Adressen, Steuersignalen und Versorgungsleitungen für Leiterkartenbaugruppen dar, die von einem Baugruppenträger, zum Beispiel mit einer Einbaubreite von 19 Zoll, aufgenommen werden. Mit Leiterkarten, die an diesen Bus angeschlossen werden, lassen sich in modularer Weise Rechner, Multiprozessorsysteme, Meßwerterfassungs- und Verarbeitungseinrichtungen für experimentelle Untersuchungen, Simulatoren, Steuer- und Überwachungsanlagen u. a. m. aufbauen. Hierbei hat der Bus wesentlichen Einfluß auf Leistungsfähigkeit und Zuverlässigkeit des Zusammenspiels der einzelnen Leiterkartenbaugruppen /1/.

Das problemlose Zusammenwirken der einzelnen Leiterkarten im Baugruppenträger setzt die präzise und eindeutige Definition der Übertragungs- und Anschlußbedingungen des Busses und bei den Leiterkartenherstellern die exakte Berücksichtigung der Busprotokolle u. ä. voraus, wenn die Vorteile eines modular aufgebauten, erweiterungs- und anpassungsfähigen „offenen" Systems zur Konfigurierung anspruchsvoller Rechner und Geräte genutzt werden sollen. Unter diesem Aspekt sind die langjährigen nationalen und internationalen Bemühungen um die Definition geeigneter Busstandards zu sehen, vor allem auch um die Akzeptanz eines Standards durch möglichst viele Leiterkarten- und Geräteproduzenten.

Bei den ersten Parallelbus-Standards für die Konfigurierung komplexer Systeme aus einzelnen Leiterkartenbaugruppen war charakteristisch, daß alle Busoperationen unter der Regie eines definierten Steuermoduls (Busmaster) abliefen, zum Beispiel die CAMAC-Rahmensteuerung im modularen Instrumentierungssystem CAMAC oder die Zentrale Verarbeitungseinheit (CPU) im Bus des Mikrorechnersystems K 1520. Mit zunehmendem Integrationsgrad der mikroelektronischen Schaltkreise und mit der dadurch möglich gewordenen Ausstattung von Leiterkarten mit lokalen Mikroprozessoren, beispielsweise zur Datenein- und Datenausgabe, wurde es notwendig, die Bussteuerung wahlweise zwischen einzelnen anfordernden (masterfähigen) Baugruppen umschalten zu können. So entstanden die Konzepte multimasterfähiger Bussysteme, die in den achtziger Jahren in Verbindung mit der Verfügbarkeit von 32-Bit-Mikroprozessoren zu einer Reihe von Busstandards geführt haben /2/, /3/, /4/, /5/, /6/. Bei der Wertung von multimasterfähigen Bussen für 32-Bit-Mikroprozessoren hat man neben den Buseigenschaften zwei wichtige Gesichtspunkte zu berücksichtigen.

- Der Vorteil eines modularen Systems mit Verfügbarkeit vieler verschiedener Leiterkarteneinschübe kommt voll zum Tragen, wenn Rechner, Verarbeitungs-, Steuersysteme u. ä. für ausgewählte Aufgabenstellungen (und demzufolge meist in nur geringer Stückzahl) und im wissenschaftlichen Gerätebau mit geringem Zeitbedarf und guter Anpassung an die Aufgabenstellung aufzubauen sind. Demgegenüber wird es im Regelfall bei Produkten, die wie Personalcomputer in großen Stückzahlen zu fertigen sind, ökonomisch günstiger sein, einen speziell auf die Rechnerkonfiguration, die eingesetzte Mikrorechner-Schaltkreisfamilie und die konstruktiven Besonderheiten optimierten Bus zu verwenden. Dieser Weg wurde unter anderem mit dem PC-Bus und dem Mikrokanal (Micro Channel Architecture – MCA) von IBM beschritten. Der Aufbau von Personal- und Arbeitsplatzcomputern mit einem multimasterfähigen Standardbus – wie dem Bus MMS-16 des A 7100 /7/ –, der elektrisch mit den Bussen Multibus I, AMS-Bus sowie I-41 kompatibel ist, dürfte mehr als Ausnahme vom üblichen Weg zu betrachten sein. In der nachfolgenden Beschreibung von Bussystemen werden die zumeist herstellerspezifischen Lösungen für kostengünstige Personalcomputer nicht berücksichtigt.
- Moderne Mikroprozessoren mit 16- und 32-Bit-Verarbeitungswortbreite zeichnen sich durch interne Parallelarbeit des Anschlußinterfaces, der Verarbeitungseinheit und weiterer Funktionen aus, so daß der Mikroprozessor praktisch ohne merkliche Pause Datentransporte und damit Busoperationen ausführt. Bei einem Multiprozessorsystem würde damit der Bus zwangsläufig zum Engpaß werden, das heißt, die Gesamtleistung eines Multiprozessorsystems würde gegenüber einem Einprozessorsystem nur relativ wenig ansteigen. Der übliche Ausweg aus dieser Situation besteht darin, dem Einzelprozessor über einen lokalen oder Nebenbus die Kommunikation mit „seinem" Speicher zu ermöglichen und den System- oder Hauptbus für die Kommunikation zwischen den Prozessoren zu nutzen. Aus dieser Konzeption heraus entstanden Bussysteme mit mehreren unabhängigen und mit unterschiedlichen Aufgaben versehenen Bussen. Ein Beispiel zeigt Bild 1 mit dem Multibus II /8/, /9/, /10/, /11/. Hier und in ähnlichen Systemen konzentrieren sich die über einzelne Hersteller hinausgehenden Standardisierungsaktivitäten vor allem auf den parallelen Systembus. Die verschiedenen Nebenbusse sind oft über die Definitionsphase nicht hinausgekommen. Andererseits ist eine sehr weitreichende Standardisierung dieser Nebenbusse auch nicht zu erwarten, wenn man bedenkt, daß sich der lokale Bus zwischen dem Mikroprozessor und seinem zugeordneten Speicher auf der Prozessorkarte selbst befinden kann.

Insgesamt läßt sich schlußfolgern, daß sich gegenwärtige Diskussionen um multimasterfähige Bussysteme für Multiprozessoranordnungen im wesentlichen auf parallele Systembusse für 32-Bit-Mikroprozessoren beziehen, deren Haupteinsatzfeld im modularen Aufbau spezieller Rechner, Datenerfassungs- und Steuersysteme und anderer Komplexe des wissenschaftlichen Gerätebaus liegt und sicherlich nicht in der Massenfertigung von „geschlossenen" Geräten zu suchen ist.

Vergleich von 32-Bit-Mikrorechnerbussystemen

Der wohl erste multimasterfähige Systembus mit nachfolgend großer Verbreitung wurde 1976 als sogenannter Multibus – später als Multibus I bezeichnet – für die Realisierung von Multiprozessorsystemen mit dem 16-Bit-Mikroprozessor 8086 definiert /12/. Die Entwicklung von 32-Bit-Mikroprozessoren führte zum Entwurf von entsprechenden 32-Bit-Bussystemen. Während der ursprüngliche VMEbus auf wahlweise 32 nutzbare Daten- und 32 Adreßleitungen erweitert wurde /13/, schlug Intel 1983 den Multibus II als einen neuen parallelen Systembus (iPSB) vor, der ebenfalls durch Zusatzbusse ergänzt wird, siehe Bild 1 /8/, /10/, /11/. In Tafel 1 sind charakteristische Eigenschaften der erwähnten Bussysteme, einiger Standards der IEEE (USA) und des japanischen TRON-Projekts zusammengefaßt, nach /2/, /3/, /4/, /5/, /14/. Für den Tobus des TRON-Projekts ist eine spätere Erweiterung auf 64 Daten- und Adreßleitungen möglich.

Zu den wichtigsten Kriterien beim Vergleich verschiedener Busstandards zählt die maximale Übertragungsrate. Die Laufzeit eines Signals auf der rund 0,5 m langen und durch die angeschlossenen Bustreiber und -empfänger kapazitiv belasteten Busleitung beträgt etwa 15 ns /1/, /15/. Berücksichtigt man die notwendigen Zeiten für das Einschwingen der Signale, für das Abklingen von Reflexionen auf der Leitung, zum Ausgleich von Laufzeitdifferenzen in den Busankoppelschaltkreisen u. ä., so ergibt sich bei vertretbarem technischen Aufwand ein minimaler Zeitbedarf für eine Busübertragungsoperation von rund 100 ns. Durch Treiber mit rampenförmig ansteigendem Signal zur Reduktion des Übersprechens und mit einer Schottky-Diode in Reihe mit dem Treiberausgang zur Verringerung der kapazitiven Busbelastung wird beim Futurebus auf eine Verringerung dieses Zeitbedarfs und damit auf eine Erhöhung der Übertragungsrate bis auf etwa 100 MByte/s orientiert /16/.

Bei einer Datenübertragung mit getrennten Adreß- und Datenleitungen ist zwar die höchste Übertragungsrate erreichbar, dennoch wird bei neueren Bussystemen die Übertragung von Adressen und Daten auf gemeinsamen Leitungen im Zeitmultiplexbetrieb bevorzugt, weil hieraus eine beträchtliche Verringerung der Zahl der Bustreiberschaltkreise und der von ihnen erzeugten Wärme resultiert. Da Daten erst nach Aktivierung des Datenwegs durch Auswertung der Adressen gelesen oder ge-

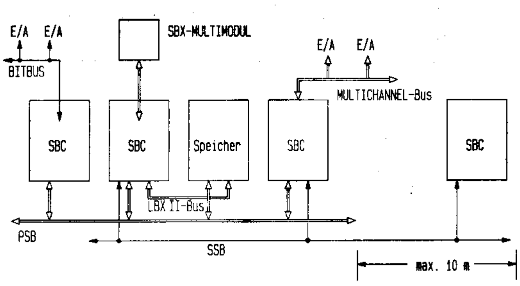


Bild 1 Konzeption des Multibus II; SBC single-board computer, Einleiterkartenrechner

Tafel 1 Vergleich von 32-Bit-Bussystemen

	VMEbus	Fastbus	Futurebus	Nubus	Multibus II	Tobus
Standard, Status	IEEE 1014	ANSI/IEEE 960 IEC 935	IEEE P896.1	IEEE 1196	IEEE 1296	TRON-Project (Japan)
Initiator, Förderer	Motorola u. a.	US-NIM-Committee	IEEE 896 WG	Texas Instr.	Intel Corp. u. a.	TRON-Project
Adreß- und Datenleitungen	24 A, 16 D, Erweiterung: 32 A, 32 D	32 AD zeitmultiplex	32 AD + 2 Tag-Leitungen, zeitmultiplex	32 AD, zeitmultiplex	32 AD, zeitmultiplex	32 AD, zeitm. (Erweiterung für 64 AD)
Datenübertragungsbreite, Bit	16; 32, 24, 8	32	32; 24, 16, 8	32, 16, 8	32, 24, 16, 8	32, 16, 8 (64)
Busübertragung (Bustakt, MHz)	asynchron	asynchron, synchron möglich	asynchron	taktsynchron (10)	taktsynchron (10)	asynchron
Paritätsbits für Busübertragung	–	1	7	1	6	5 (9)
Adreßräume	64	2	1	1	4	4, 4, reserviert
geographische Adressierung	–	5 Bit ID	5 Bit ID	4 Bit ID	ID-Zuordnung bei Initialisierung	?
maximale Übertragungsrate, MByte/s	33	Block: 70 Einzelwort: 35	60 ... 100	Block: 37,5 Einzelwort: 20	Block: 40 Datenmessage: 32 Einzelwort: 20	Einzelwort: 50 (100)
Massage-passing	–	für Interrupt	für P896.2 vorgesehen	–	ja	für Interrupt
Interruptleitungen	7	1	–	–	–	2
virtuelle Interrupts	–	für Master	ja	ja	Message-passing	Messages
Arbitration	zentral 4 Ebenen	verteilt	verteilt asynchron	verteilt taktsynchron	verteilt taktsynchron	verteilt
Anzahl aktiver Busleitungen	107	60	67	46	67	75 (115)
Steckverbinder	1 × 96 (32 Bit: 2. Steckverbinder)	1 × 130	1 × 96* IEC 603-2	1 × 96 IEC 603-2	1 × 96 IEC 603-2	1 × 96 IEC 603-2 (64 Bit: 2. St.)
Leiterkartengröße, mm² (Eurocard IEC 297-3)	233,35 × 160 100 × 160	366,7 × 400	366,7 × 280 233,35 × 280	366,7 × 280	233,35 × 220 100 × 220	233,35 × 160 100 × 160 (366,7 × 160)
[illegible]st zugeordnete [illegible]eckplätze	1 (für Arbitration)	–	–	–	1 (für CSM)	–
[illegible]satzbusse	VMXbus (lokal) VMSbus (seriell) I/O channel VSB (IEEE P1096 als Subsystembus)	serieller Bus vorgesehen	serieller Bus vorgesehen	–	iLBX (lokal) iSSB (seriell) iSBX (E/A-Erweiterung, IEEE P959) MULTICHANNEL BITBUS	Toxbus (lokal)
Unterstützung durch Treiber- und Bussteuerschaltkreise	Transceiver, Steuerschaltkreise der Mikroprozessorfamilie 68000, VMEchip	Transceiver für Bus und Kabelsegment	Transceiver	Transceiver, Steuerschaltkreise	Transceiver, BAC 84110, MIC 84120, Message-passing-Koprozessor 82389	Steuerschaltkreis geplant

schrieben werden können, bringt die zeitlich gestaffelte Übertragung von Adressen und Daten im allgemeinen keine gravierenden Nachteile.

Beim Busübertragungsprotokoll wird zwischen asynchroner und synchroner Übertragung unterschieden. Während sich die asynchrone Übertragung mit Hilfe von Anmeldungs- und Quittierungssignalen automatisch an die jeweils höchstmögliche Übertragungsgeschwindigkeit anpassen kann, zeichnet sich eine synchrone Übertragung durch ein einfacheres Businterface und eine geringere Störempfindlichkeit aus, wenn die Signalpegel nach dem Einschwingen taktsynchron bewertet werden. Diese synchrone Busübertragung wird beispielsweise beim Multibus II durch ebenfalls taktsynchron abgetastete Handshake-Signale gesteuert. Bei einigen der in Tafel 1 aufgeführten Bussysteme sind die Daten und die Steuersignale durch ein oder mehrere Paritätsbits zusätzlich gesichert.

Bei den in Tafel 1 dargestellten Bussystemen werden für die Arbitration, das heißt für die Zuteilung der Bussteuerung zu einem von mehreren anfordernden Mastermodulen, zwei Möglichkeiten, jeweils mit Berücksichtigung von Prioritäten, genutzt: die zentrale Arbitration mit Erteilung der Buszugriffsberechtigung über spezielle Busleitungen an den „gewinnenden" Master und die verteilte Arbitration, bei der die Arbiter der anfordernden Module gemeinsam die aktuelle Zugriffsberechtigung ermitteln.

Wenn in einem Multiprozessorsystem die Interruptanforderung eines bestimmten Prozessors zu einem ausgewählten Modul übertragen werden soll, kann dies bei nur wenigen Interruptquellen mit Hilfe von separaten und entsprechend angeschlossenen Interruptleitungen gelöst werden. Diese Methode versagt jedoch, falls eine Interruptanforderung eines von N Modulen zu einem der anderen (N-1) Module zu übermitteln und N größer als etwa 8 bis 10 ist. Eine wirkungsvolle Lösung dieser Aufgabe wurde mit dem Prinzip der virtuellen Interrupts beim Multibus II geschaffen.

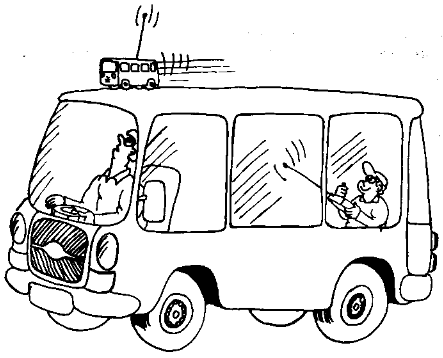
Die in Tafel 1 genannten 32-Bit-Bussysteme haben unterschiedliche praktische Bedeutung erlangt. Der VMEbus ist international weit verbreitet, wird von vielen Herstellern unterstützt und bildet die Grundlage zum Aufbau von Rechnern und Steuerungen sowie im wissenschaftlichen Gerätebau. Die Domäne des Fastbusses liegt in der Instrumentierung von Experimentanordnungen für die Hochenergie- und Elementarteilchenphysik, wo oftmals im Verlauf weniger Mikrosekunden die Ausgangssignale von Zehntausenden gleichartiger Detektoren erfaßt werden müssen. Die seit mehr als 10 Jahren laufenden Diskussionen zur Spezifikation des Futurebusses sind von den Entwicklungen anderer Bussysteme beeinflußt worden, haben aber auch zur Gestaltung solcher Standards wie Multibus II beigetragen. Fehlende industrielle Akzeptanz des Futurebusses – erkennbar auch am Fehlen geeigneter Interface-Steuerschaltkreise – verhinderte bislang einen merklichen Einsatz. Der Nubus als „einfacher" Bus hat bisher zum Aufbau modular strukturierter Multiprozessorsysteme offenbar keine Anwendung gefunden. Der parallele Systembus iPSB des Multibus II, dessen Architektur gegenüber den anderen genannten Bussen durch einige weiterführende Besonderheiten gekennzeichnet ist, setzt sich international immer mehr durch. In /9/ und /17/ ist eine Reihe von Multibus II-Modulen als Verarbeitungseinheiten (mit den Mikroprozessoren 80286/386/486), als Speicher, Kommunikations-, Bildschirm- und Peripheriesteuerungen sowie als Ein- und Ausgabemodule aufgeführt.

Gegenwärtig wird zu Forschungs- und Standardisierungsarbeiten für einen Multiprozessorbus mit einer Übertragungskapazität von 1 GByte/s aufgerufen /18/. Dieser „Superbus" mit der Bezeichnung Scalable Coherent Interface (SCI) soll gegenüber den besten derzeit spezifizierten Bussen eine um mindestens eine Größenordnung höhere Übertragungsleistung aufweisen, unter anderem durch einen 250-MHz-Übertragungstakt und unidirektionale Übertragungsleitungen. Es beginnen auch Diskussionen zu einem sogenannten Futurebus-Plus, der eine Übertragungsbreite von bis zu 256 Bit aufweisen soll und der für den Datenaustausch mit schnellen Massenspeichern einsetzbar ist.

Multibus II

In der „Philosophie" des Multibus II stellt jede Leiterkarte für sich einen funktionsfähigen Mikrorechner dar, der Anwendersoftware unter der Regie eines Betriebssystems nutzt oder als Steuerung des Datenaustauschs mit externen Einheiten, als Grafikprozessor o. ä. arbeitet (siehe Bild 2). Der Parallelbus iPSB dient damit im wesentlichen nur zum Transfer von Nachrichten (messages) zwischen den einzelnen Modulen des Systems, das heißt, der iPSB läßt sich als lokales Netz betrachten, dessen Ausdehnung auf die Breite des Baugruppenträgers beschränkt ist. Der Multibus II als IEEE-Standard 1296 /8/, /10/ erhielt 1988 vom Technischen Komitee SC47B des IEC einen internationalen Status. Kurzbeschreibungen sind in /6/ und /19/ und mit Beachtung von Software- und Systemaspekten in /11/ zu finden.

Die wesentlichen Eigenschaften des Multibus II betreffen den Nachrichtenaustausch (message-passing) und die Unterstützung der Systemkonfigurierung und -kontrolle. Bei der Message-passing-Kommunikation erfolgt der Datentransfer auf dem parallelen



Externbus *Zeichnung: Frank Steger*

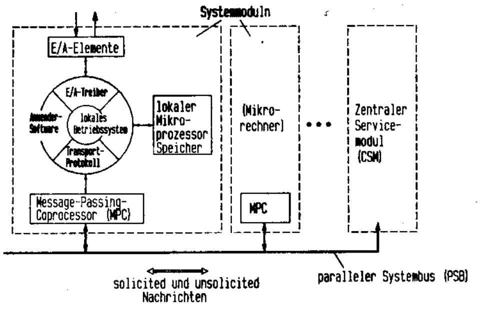
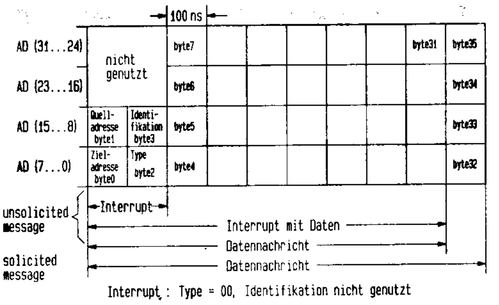


Bild 2 Systemmodule des Multibus II am parallelen Systembus iPSB

Bild 3 Nachrichtenformate bei Message-passing-Kommunikation auf dem parallelen Systembus iPSB



Systembus durch Datenpakete (Nachrichten) mit einer Länge von 36 oder 32 Byte. Die *nicht angemeldeten Nachrichten* (unsolicited messages) werden als Interruptanforderungen an einen Zielmodul (Zieladresse) oder an alle Module (broadcast) gerichtet und enthalten neben den Adressen (jeweils 1 Byte für Quelle und Ziel) ein maximal 28 Byte umfassendes Parameterfeld (Bild 3). Durch diese virtuellen Interrupts wird das erwähnte N x (N-1)-Problem der Interruptanforderungen bei mehreren interruptfähigen Einheiten elegant gelöst. Mit nicht angemeldeten Nachrichten läßt sich ferner die Übertragung angemeldeter Nachrichten vorbereiten. *Angemeldete Nachrichten* (solicited messages) weisen neben der Quell- und der Zieladresse und dem bis zu 32 Byte langen Datenfeld zusätzliche Typangaben auf. Nach Anforderung von Pufferspeicherkapazität im Empfangsmodul und Bestätigung durch den Empfänger wird der Datenblock unter Regie der Message-passing-Koprozessoren der betreffenden Module übertragen. Wesentlich ist, daß hierbei die lokalen Mikroprozessoren nicht beteiligt sind und daß auch die Aufteilung größerer Datenfelder in 32-Byte-Nachrichten oder die Zusammenfügung übertragener Nachrichten von den Message-passing-Koprozessoren selbständig abgewickelt wird. Die Message-passing-Übertragung erfolgt damit unabhängig von den lokalen Mikroprozessoren und vom Nutzerprogramm, und sie ist damit auch nicht an bestimmte Mikroprozessortypen gebunden. Der als integrierter Schaltkreis entwickelte Message-passing-Koprozessor MPC 82389 /20/ ermöglicht somit eine einfache und wenig Leiterkartenfläche beanspruchende Kopplung zwischen Leiterkarte und iPSB.

Mit der sogenannten geographischen Adressierung kann nach Initialisierung des Systems durch einen zentralen Service-Modul (CSM) des Multibus II jeder Systemmodul über die Steckplatzposition aufgerufen werden, ohne daß in den Modulen hierfür Schalter oder Drahtbrücken zu verändern sind. Der im Systemmodul vorhandene Interconnect-Registerblock mit einer Länge von 512 Byte enthält durch den Hersteller eingespeicherte Angaben zum Modultyp u. ä. Die Konfigurationsregister des Interconnect-Speichers können im Interconnect-Adreßraum durch das Betriebssystem mit Einstellparametern für die gewünschte Modulbetriebsweise geladen werden. Neben der Vermeidung manueller Bedien- und Einstellfehler ist es vorteilhaft, daß die Konfigurierung des Systems vom Betriebssystem – einschließlich des Aktivierens von Reserve-Modulen nach dem Erkennen fehlerhaft arbeitender Module – ermöglicht wird. Im Zusammenhang mit der Busspezifikation muß auch auf modulresidente Testsoftware verwiesen werden (built-in self test – BIST), die im Modul unter Steuerung eines 8-Bit-Einchipmikrorechners abläuft. Sie erlaubt nach Einschalten der Betriebsspannung einen Selbsttest und dient auch einer möglichst hohen Systemzuverlässigkeit.

Ausblick

Die Wahl eines „besten" Bussystems aus der Vielfalt der definierten und eingeführten Bussysteme ist wegen einer Reihe sich teilweise widersprechender Forderungen nicht möglich. Von wesentlicher Bedeutung ist jedoch, ob geeignete Busankoppel- und Bussteuerschaltkreise zur Verfügung stehen und ob sich der Anwender auf ein möglichst breites, im internationalen Maßstab eingeführtes Modulsystem stützen kann. Unter diesen Voraussetzungen können modulare Systeme mit hoher Leistungsfähigkeit, mit Nutzung vorhandener Erfahrungen und Software und mit Anpassungsfähigkeit an wechselnde Aufgabenstellungen bei vertretbarem Zeitaufwand geschaffen werden. Dies steht vor allem im wissenschaftlichen Gerätebau und bei Rechnern und Steuerungen für Einzelanwendungen im Vordergrund.

Allgemein wird eingeschätzt, daß trotz auftretender Kompatibilitätsprobleme bei VMEbus-Modulen unterschiedlicher Hersteller der gegenwärtig dominierende VMEbus für die angedeuteten Einsatzgebiete auch bis in die neunziger Jahre hinein die Hauptrolle spielen könnte. Ab Mitte der neunziger Jahre wird eine Vorrangstellung des Multibus II vorausgesagt, dessen Architekturmerkmale denen des VMEbus überlegen sind. Die nicht zu unterschätzende Kompliziertheit aller Bussysteme und die Einbindung in Betriebssysteme lassen eine intensive Mitarbeit an den international betriebenen Normungsaktivitäten als vordringlich erscheinen, da nur so rechtzeitig eine optimale nationale Orientierung und die entsprechende Berücksichtigung der Entwicklungen in der gerätebauenden Industrie ermöglicht wird.

Literatur

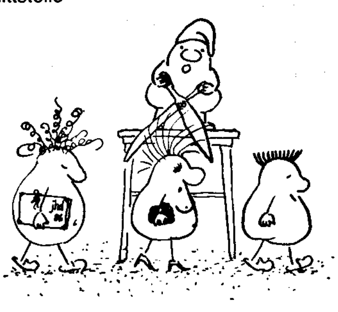
/1/ Gustavson, D. B.: Computer buses – a tutorial. IEEE Micro 4 (1984) 4, S. 7
/2/ Smolin, M.: Micro standards. IEEE Micro 8 (1988) 3, S. 5
/3/ Müller, K. D.: Comparison and status of 32 bit backplane bus architectures. IEEE Trans. Nucl. Sci. 32 (1985), S. 262
/4/ Borrill, P. L.: Objective comparison of 32-bit buses. IEEE Micro 5 (1985) 6, S. 71; Microprocessors and microsystems 10 (1986), S. 94
/5/ Gustavson, D. B.: High performance data buses – progress and evolution. IEEE Trans. Nucl. Sci. 35 (1988), S. 326
/6/ Meiling, W.; Fülle, R.: Mikroprozessoren und Mikrorechner. Berlin: Akademie-Verlag 1988
/7/ Junge, S.; Keller, D.: Das Mikrorechnermodulsystem 16 und sein Einsatz im Arbeitsplatzcomputer robotron A 7100. Neue Tech. Büro, Berlin 29 (1985) 3, S. 81
/8/ Intel Corp.: MULTIBUS II bus architecture specification, handbook. 146077-B, 1983 und spätere Ausgaben
/9/ Intel Corp.: MULTIBUS II product data book. 280055-003, 1986
/10/ IEEE Std. 1296, MULTIBUS II: A full-feature 32-bit backplane bus. 1987
/11/ Furrer, F. J.: System- und Software-Architektur des Multibus II. Elektronik, München 37 (1988) 21, S. 78, 38 (1989) 2, S. 75
/12/ Johnson, J. B.; Kassel, S.: The MULTIBUS design guidebook – structures, architectures, and applications. New-York: McGraw-Hill 1984
/13/ Fischer, W.: IEEE P1014 – a standard for the high-performance VME bus. IEEE Micro 5 (1985) 1, S. 31
/14/ Sakamura, K.; Sano, R.; Honma, R.: Introducing Tobus. The system bus in the TRON architecture. IEEE Micro 8 (1988) 2, S. 47
/15/ Wilson, R.: The physics of driving backplane buses. Microprocessors and microsystems 10 (1986), S. 61
/16/ Balakrishnan, R. V.: The proposed IEEE 896 Futurebus – a solution to the bus driving problem. IEEE Micro 4 (1984) 4, S. 23
/17/ European Multibus Group EMG: Multibus I, AMS bus, Multibus II, Product guide 89/90. München, 1989
/18/ Gustavson, D. B.: IEEE P1596, a Scalable Coherent Interface for gigabyte/sec multiprocessor applications. IEEE Trans. Nucl. Sci. 36 (1989), S. 811
/19/ Schwertfeger, H.-J.; Krüger, W.: Ein- und Mehrprozessorsysteme mit Multibus-Architektur. Mikroprozessortechnik, Berlin 2 (1988) 3, S. 83
/20/ Intel Corp.: 82389 Message Passing Coprocessor. A MULTIBUS II bus interface controller (Advance information). Microprocessor and peripheral handbook 1989, S. 6.150

✉ KONTAKT ☎

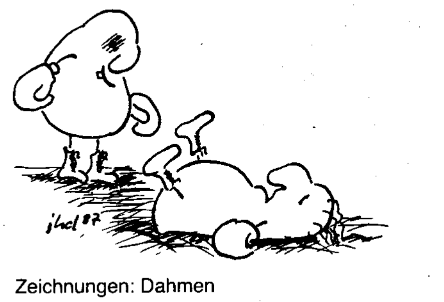
Technische Universität Dresden, Sektion Physik, Mommsenstraße 13, Dresden, 8027; Tel. 4 63 50 76

Kleines Lexikon der Mikrorechentechnik

S
wie Schnittstelle



T
wie Timeout



Zeichnungen: Dahmen

Die Dateihierarchie bei größeren Softwareprojekten unter Unix

Peter Lange, Dieter Büttner, Dresden

Bei der Entwicklung und Wartung größerer Softwareprojekte entstehen besondere Anforderungen an die Technologie, mit der die Programme hergestellt werden. Es wird im folgenden eine Dateihierarchie für das Betriebssystem Unix vorgestellt, die die Softwareentwicklung für solche Projekte unter Verwendung der typischen Hilfsmittel von Unix unterstützt.

Damit ist eine Softwareherstellung unter den folgenden Randbedingungen möglich:

- Das Projekt soll von vielen Anwendern genutzt werden, die das Projekt auch selbständig weiterentwickeln können.
- Es sollen unterschiedliche Versionen des Projektes gleichzeitig nebeneinander existieren können.
- Mehrere Bearbeiter sollen gleichzeitig an der Herstellung der Software beteilgt sein können.
- Das Programmsystem soll fertig generiert ausgeliefert oder vom Nutzer selbst generiert werden können.
- Die Fehlersuche und Wartung muß auch beim Nutzer möglich sein.
- Die Zuordnung der Programme zu ihren Autoren soll sichtbar bleiben.
- Die einzelnen Komponenten sollen relativ isoliert vom Gesamtsystem bearbeitet werden.
- Es sollen die Unix-Tools verwendet werden.
- Die typische Arbeitsweise in Unix muß erhalten bleiben.

Um diesen Forderungen gerecht zu werden, muß eine Hierarchie von Dateiverzeichnissen, wie sie als Beispiel in Bild 1 dargestellt ist, entwickelt werden. In diesen Dateiverzeichnissen befinden sich sämtliche zu einem Projekt gehörenden Dateien. Es gibt zwei Typen von Dateibäumen:

1. Der Baum ab Directory *relis* enthält alle Dateien zur Generierung des Gesamtsystems RELIS und bildet die Umgebung für die Systemgenerierung.
2. Der Baum ab Directory *otto* enthält die Dateien für die Arbeit des Programmierers Otto. Dieser Teilbaum stellt die Umgebung für einen Programmierer dar.

Die Entwicklerumgebung des Programmierers

Der wesentliche Unterschied zwischen dem Dateibaum des Gesamtsystems und dem Baum für die Projektbearbeiter ist ein Directory *sccs* für Quelldateien. Die Quellen für alle Programmversionen, die ein Bearbeiter erstellt, werden in Unterverzeichnissen des Verzeichnisses *sccs* mit Hilfe des unter Unix verfügbaren SCCS (Source Code Control System) verwaltet.

Es wurden folgende Unterverzeichnisse aufgebaut:

../sccs/txt enthält alle vom Entwickler zu erstellenden Texte (Anwendungsbeschreibung, Entwicklungstechnologie usw.) als NROFF-Quellen (Suffix:.nq)

../sccs/dat Alle vom Entwickler benutzten Dateien, die zur Erprobung seiner Programme notwendig sind, und von ihm als Beispiel- oder Testdateien in eine Version eingebracht werden sollen, sind in diesem Verzeichnis abgelegt.

../sccs/h speichert alle vom Entwickler bereitzustellenden Definitionsdateien (Include-Files);

../sccs/src beinhaltet alle C-Quellen (C-Funktionen und C-Programme) sowie für das Make-File des Entwicklers mit den Vorschriften zum Bilden von Dokumentationen zu seinen Komponenten, von Objekt-Modulen für die RELIS-Bibliothek sowie für lauffähige C-Programme und Shell- bzw. C-shell-Prozeduren;

../sccs/csh enthält alle vom Entwickler bereitzustellenden Shell- und C-shell-Prozeduren.

Zur Programmentwicklung hat ein Entwickler für jede Programmversion ein separates Directory (hier *v1* für die Entwicklung der Version 1). In dem als *lokale Entwicklerumgebung* bezeichneten Teilbaum unterhält er den durch das SCCS für die jeweilige Version bereitgestellten Quellcode, und dort findet er die in dem als *Testumgebung* bezeichneten Teilbaum auch die Systemkomponenten, die er für seine *private* Generierung zu Testzwecken benötigt. Jeder Entwickler kann davon ausgehen, daß er in der Testumgebung sämtliche bisher für die entsprechende Version freigegebenen Softwarebestandteile vorfindet, denn diese Umgebung wird für ihn bei jeder neuen Generierung innerhalb einer Version aktualisiert. Die Unterverzeichnisse der *Testumgebung* haben folgenden Inhalt:

lib Die aktuelle RELIS-Bibliothek librelis.a wird aus der Umgebung der entsprechenden Version des Programmsystems hierher kopiert. Dabei erhält der Entwickler *Schreib*erlaubnis für diese Bibliothek, wodurch das Ändern von Objektmodulen oder das Einbringen neuer Objektmodule aus der lokalen Entwicklerumgebung möglich wird.

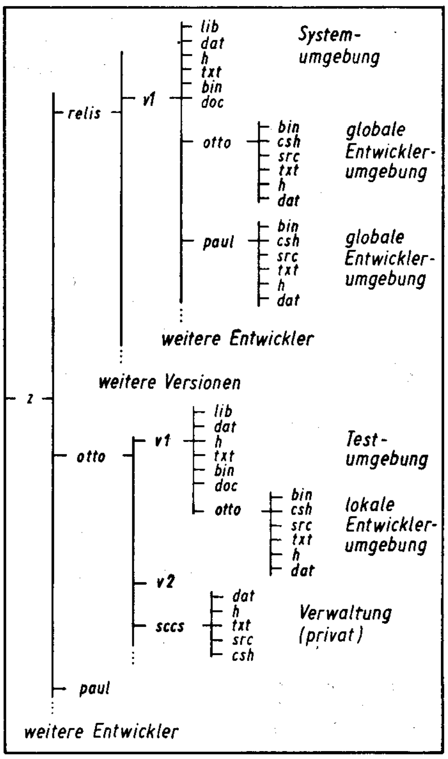


***Bild 1* Dateihierarchie**

dat Sämtliche Dateien aus den Dat-Directories aller Entwickler, die in eine Generierung eingeflossen sind, werden hierher kopiert. Damit ist beim *privaten* Testen der Zugriff auf alle bisher vorhandenen Datenbestände gesichert. Die Zugriffsrechte erlauben es dem Entwickler, diese Dateien zu ändern, um gegebenenfalls seine Tests zu erweitern.

h Hier werden sämtliche Definitionsdateien (Include-Files) für die Generierung durch den Entwickler bereitgestellt. Im Gegensatz zu den Verzeichnissen *lib* und *dat* erfolgt hier ein Kopieren auch mit Schreibberechtigung nur für diejenigen Dateien, die eben jener Entwickler für die aktuelle Version bereitgestellt hat. Die Include-Files aller anderen Entwickler finden sich hier als Ergebnis des Linkens wieder, und dem Entwickler ist das Ändern dieser Dateien verboten!

bin Das oben für die Incluce-Files bezüglich Bereitstellung Gesagte gilt auch für das Directory *bin.* Alle C-Programme und Shell- bzw. C-shell-Prozeduren anderer Entwickler stehen in der *lokalen Testumgebung* zur Verfügung. Das Ändern ist wiederum nur für die eigenen Komponenten möglich! Der Pfad zu diesem Directory ist für das Testen in die Environment-Variable $PATH einzugliedern.

txt Ist der Entwickler an der Erarbeitung von Texten allgemeiner Art beteiligt, so hat er die Möglichkeit, die mit **nroff** formatierten Texte zu begutachten.

Die Generierung der Komponenten der Entwickler wird durch Make-File – eine Handlungsanweisung für das Unix-Tool Make – gesteuert. Damit gibt der Programmentwickler an, wie unter Verwendung der Testumgebung seine Programmkomponenten gebildet werden.

Diese Generierung verläuft in folgenden Schritten:

make h

Aus dem Subdirectory *h* einer Entwicklerumgebung werden alle Include-Files in das Subdirectory *h* einer *Testumgebung* kopiert. Gleichnamige Dateien werden dabei überschrieben. Damit wird gewährleistet, daß beim nachfolgenden Kompilieren von C-Quellen die aktuellen Include-Files einer Version zusammen mit den eigenen Include-Files verwendet werden.

make lib

In diesem Schritt werden neue bzw. geänderte Bibliotheksfunktionen in die RELIS-Bibliothek *librelis.a,* die als Kopie im Subdirectory *lib* einer Testumgebung vorhanden ist, aufgenommen. Mit dem Compiler **CC** werden Objekt-Module erzeugt, die mit dem Archivar **AR** in librelis.a eingetragen werden.

make dat

Aus dem Subdirectory *dat* einer Entwicklerumgebung werden alle Daten-Files in das Subdirectory *dat* einer Testumgebung kopiert. Gleichnamige Dateien werden dabei überschrieben.

make txt

Aus dem Subdirectory *txt* einer Entwicklerumgebung werden alle als *NROFF*-Quellen vorhandenen Textdateien mit NROFF forma-

tiert und im Subdirectory *txt* einer Testumgebung abgelegt.

In der folgenden Darstellung werden die Abläufe der Generierungsschritte *make csh* und *make bin* an Hand von Bild 2 näher erläutert. Darin werden Shell- und C-shell-Prozeduren für eine *private Testumgebung* bereitgestellt. Des weiteren werden aus C-Quellen unter Einbeziehung der vorher aktualisierten Include-Files und der RELIS-Bibliothek die lauffähigen C-Programme erzeugt.

make csh (1)

Hierbei werden *link*-Dateien aller Shell- und C-shell-Prozeduren, die sich in der *Entwicklerumgebung* befinden, in den *Bin*-Verzeichnissen der Entwicklungs- und der Testumgebung angelegt.

make bin (2)

Als nächstes werden nach den Bildungsvorschriften aus C-Quellen unter Einbeziehung der Include-Files (*) und der RELIS-Bibliothek lauffähige C-Programme erzeugt und physisch im Subdirectory für C-Quellen (3) abgelegt. Bei erneuten Generierungsläufen kann *Make* somit feststellen, welche C-Programme noch aktuell sind und demzufolge nicht neu erzeugt werden müssen. Alle nach den Vorschriften neu erstellten C-Programme werden als Link-Dateien (4) in die Verzeichnisse der *Entwicklerumgebung* und der lokalen *Testumgebung eingetragen* (Kompatibilität!).
Sind diese Schritte getan, stehen dem Entwickler nunmehr alle Komponenten einer gültigen Version zur Verfügung, und er hat zum Testen seine neuen Komponenten hinzugefügt. Nach Abschluß dieser Testarbeiten schließt sich der Kreis der Software-Entwicklung mit der Freigabe der erfolgreich getesteten Komponenten und ihrer Übergabe in die *Verwaltungsumgebung*, wodurch sie bei einer neuen Generierung zum Bestandteil dieser Version werden.

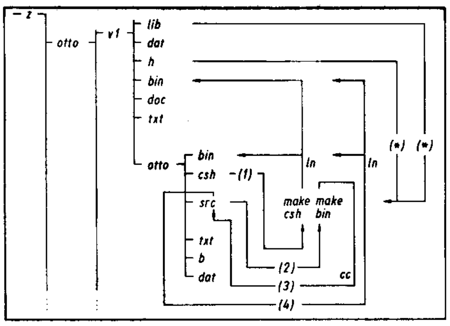


Bild 2 Ablauf der Generierung

Die Generierung des Gesamtsystems

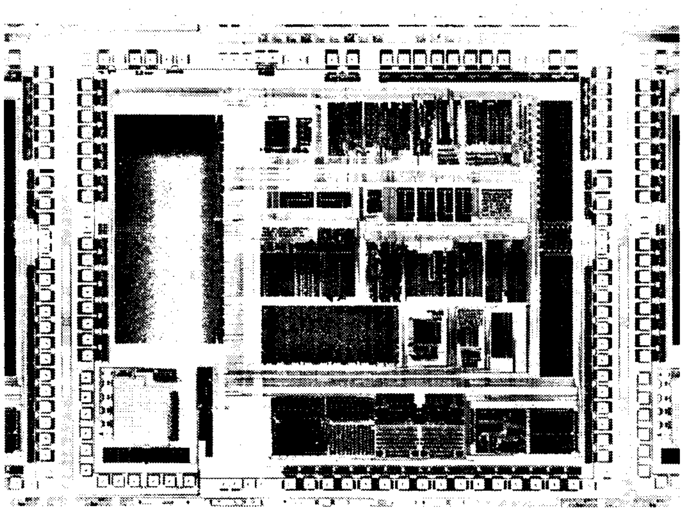
Die gleiche Struktur der Umgebung des Gesamtsystems und der Umgebung für die Programmentwickler vereinfacht die Programmentwicklung wesentlich:
Aus den SCCS-Directories der Bearbeiter werden alle Quellen für die zu generierende Version mit Hilfe von SCCS in die für jeden Programmhersteller vorhandenen Directories *csh, src, h* und *dat* geholt. Dadurch stehen auch die Make-Files eines jeden Bearbeiters für die Generierung des Systems zur Verfügung.
Die globale *Entwicklerumgebung* entspricht der lokalen *Entwicklerumgebung*. Der einzige Unterschied besteht darin, daß die Quellfiles nicht editierbar sind. Es wird von dem Grundsatz ausgegangen, daß ein Quellfile für eine Version nur an genau einer Stelle editiert werden sollte – nämlich in der *lokalen Entwicklerumgebung* dieser Version. Deshalb ist die *Systemumgebung* von der globalen Entwicklerumgebung aus über die gleichen relativen Pfade erreichbar, wie die Testumgebung des Entwicklers aus der lokalen Entwicklerumgebung. So kann also die Systemgenerierung mit den gleichen Make-Files ablaufen, mit denen während der Entwicklungs- und Testphase in der lokalen Entwicklerumgebung und der Testumgebung gearbeitet wurde. Kurz gesagt: Die Systemgenerierung besteht gleichfalls aus dem Aufruf von *make h* für alle globalen Entwicklerumgebungen, dann gefolgt von den Befehlen *make lib, make bin, make dat* und *make txt* für alle globalen Entwicklerumgebungen.

⊠ KONTAKT ☏

Forschungsinstitut für Medizinische Diagnostik Dresden, Abteilung Laborinformationssysteme, Karl-Marx-Straße 3, Dresden, 8080; Tel. 58 31 45

vorgestellt

Microcontroller SAB 80C166



Als nach eigenen Angaben einziger europäischer Hersteller stellt Siemens jetzt einen in 1,2-µm-CMOS-Technologie selbstentwickelten 16-Bit-Microcontroller vor, der in seiner Kombination von Leistungsfähigkeit und Flexibilität Maßstäbe setzen soll. Erste Kundenmuster sollen im I. Quartal 1990 verfügbar sein, die Serienfertigung beginnt Ende 1990.

Für die Steuerung komplexer Prozesse gewinnen Microcontroller – oft auch als Einchipmikrorechner bezeichnet – zunehmend an Bedeutung, wobei die als Embedded Control bezeichneten Anwendungen, zum Beispiel in der Automobiltechnik, für Steuerungsaufgaben in der Industrie und in der Datenverarbeitung, ganz spezielle Anforderungen stellen: Im Vordergrund stehen die arithmetische Leistungsfähigkeit, hochsprachenorientiertes Datenhandling, kurze Interrupt-Antwortzeiten mit Priorisierungsmöglichkeit sowie weitgehend autarke Peripherie.
Da bisherige Controller diesen Anforderungen nur bedingt gewachsen waren, hat Siemens einen 16-Bit-Rechner als neuartige Symbiose aus Prozessor und Controller entwickelt. Funktional besteht der Baustein aus einem Hochleistungs-Mikroprozessor, einem programmierbaren Interrupt-System und einer integrierten, weitgehend selbständigen modularen Peripherie.
Der *Prozessor* wird aus dem eigentlichen CPU-Kern, 1 KByte RAM, 8 KByte ROM und einem Taktgenerator gebildet.
Die *CPU* zeichnet sich durch eine der RISC-Technik vergleichbare Rechnerleistung aus, die über 90 Prozent aller Befehle in 100 ns abarbeitet. Weitere Stärken sind umfassende Bitverarbeitung, hochsprachenorientierte Adressierungsarten sowie die arithmetische Leistungsfähigkeit (z. B. 16 × 16 Bit in 500 ns, 32/16 Bit in 1000 ns). Durch Register-Banking können im internen RAM fast beliebig viele, maximal 16 × 2 Byte große General-Purpose-Register definiert werden.
Beim *Interrupt-System*, dem Kernstück der Ereignisverwaltung, kann beim SAB 80C166 jeder Interruptquelle eine von 16 Prioritätsebenen zugeordnet werden. Die hohe Echtzeit-Leistungsfähigkeit zeigt sich in der kurzen Interrupt-Antwortzeit von maximal 400 ns.
Auch der *Peripheral Event Controller* trägt zur schnellen Reaktionsfähigkeit des Systems bei, weil er die CPU von interruptgetriebenen Datentransportaufgaben entlastet (DMA-Prinzip).
Die *Peripherie-Module*, die Schnittstellen zur eigentlichen Applikation, bestehen aus einem 10-Bit-A/D-Wandler, einer 16-Kanal-Capture/Compare-Einheit, 5 General-Purpose-Timern, zwei unabhängigen USARTs, 76 I/O-Portlines und einem Watchdog-Timer.
Zur Hard- und Softwareentwicklung bietet Siemens ein umfangreiches Tool-Paket, bestehend aus C-Compiler, Assembler, Linker, Locater, Software-Simulator, und einen ab Mitte 1990 verfügbaren Echtzeit-Emulator.

MP-We

Turbo-Pascal 5.5

Der Einstieg in die objektorientierte Programmierung

Uwe Schulze, Berlin

Sie haben die Wahl: Alles wie gehabt (bei vollständiger Kompatibilität zu 5.0) oder objektorientiert. Doch damit nicht genug: Es ist möglich, objektorientierte und traditionelle (imperative) Programmierungstechniken gemischt zu verwenden. Die Verfechter objektorientierter Programmierung werden dieser Ankündigung kritisch gegenüberstehen – ist doch der direkte Zugriff auf Daten beispielsweise bei Smalltalk nicht möglich.
Der mit Turbo-Pascal 5.5 (und übrigens auch Quick-Pascal 1.0) eingeschlagene Weg stellt einen erfolgversprechenden Versuch dar, die unbestrittenen Vorteile objektorientierter Programmierung einem breiten Anwenderkreis zugänglich zu machen und für die tägliche Arbeit zu nutzen. Als Vorbild dafür sah man Smalltalk für die allgemeinen Prinzipien und C++ für die Einbindung in prozedurale Sprachen. Für die *C-Gemeinde* steht mit Zortech C++ ein ähnlich einfaches Werkzeug wie Turbo-Pascal zur Verfügung.
Der sogenannte *objektorientierte Ansatz* wurde Ende der 70er Jahre im *Xerox Palo Alto Research Centre* entwickelt und mit Smalltalk-80 ein darauf basierendes Programmiersystem der Öffentlichkeit vorgestellt.
Die Idee der Erweiterung konventioneller Programmiersprachen um objektorientierte Eigenschaften kommt nicht von ungefähr. Vielmehr ist sie eine Folge von bestimmten Problemen der Softwareentwicklung der letzten Jahre.
Als Beispiel wollen wir uns die Verwaltung von dynamischen Listen ansehen. Bei der Arbeit mit Listen sind stets die gleichen Operationen auszuführen: Anfügen eines Elements, Löschen eines Elements, Suchen nach einem bestimmten Element etc.
Allgemein nutzbare Routinen für diesen Zweck zu schreiben fällt schwer, da ja die einzelnen Listen eine unterschiedliche Struktur besitzen. Entweder wir entwerfen für jede Liste neue Prozeduren (was gängige Praxis ist) oder die Routinen werden mit Parametern versehen. Spätestens bei einer Suchroutine zeigen sich aber deutlich die Nachteile dieses Vorgehens: Der Typ des Suchschlüssels kann unterschiedlich sein, was geschieht mit doppelten Schlüsseln etc. Und was passiert, wenn der Nutzer der Routine zusätzliche Ideen einbringen will, die nicht vorgesehen sind? Hier zeigen sich zwei grundsätzliche Probleme der Programmentwicklung:

- Parameterbeladene Prozeduren oder verschiedene Prozeduren für ähnliche Probleme
- Fehleranfällige Änderungen ziehen sich durch die Programme; liegen keine Quellen vor, bleibt nur Neuschreiben.

Wir wollen versuchen, diesen Problemen mit einem objektorientierten Ansatz zu begegnen. Die Idee liegt in der Zuordnung von Prozeduren (im objektorientierten Sprachgebrauch Methoden) zu den Daten. Dazu wurde der Record-Typ erweitert. Ein einfacher Listentyp hätte damit folgendes Aussehen:

```
type
listeptr = ^liste_typ;
liste_typ = object
key          : string;
next         : listeptr;
constructor init;
procedure add(liste,elem: listeptr);
procedure delete(liste: listeptr);
procedure show(liste: listeptr);
destructor done;
end;
```

Es ist zu erkennen, daß die Prozeduren an die Daten gekoppelt sind; wie die Prozeduren programmiert sein können, ist in Bild 1 zu sehen. Mit *add* soll ein übergebenes Listenelement nach dem ersten eingefügt werden, *delete* löscht das vorderste eingefügte Element und *show* zeigt alle Schlüssel der Liste – und das sind vorerst die einzigen Datenfelder – an. Wird mit Units gearbeitet, so kann der Prozedurkörper im Implementation-Teil stehen, er ist zugänglich, wenn das Objekt nur im Interface-Teil deklariert ist.
Für den Aufbau einer Liste wird zuerst eine Variable des Objekttyps (im objektorientierten Sprachgebrauch Instanz) erzeugt:

var liste: listeptr;

Die Erzeugung eines ersten Listenelements kann mit

New(liste);

erfolgen. Anschließend muß eine Initialisierung mit

liste.init;

vorgenommen werden. Zum einen erfordert unsere programmtechnische Umsetzung, daß der Zeiger auf das nächste Element mit NIL initialisiert wird zum zweiten ist die Initialisierung eines Objektes immer notwendig, wenn mit virtuellen (das heißt später veränderbaren) Methoden gearbeitet wird.
Mit dem Ziel, die *Show*-Routine später eventuell zu ändern, wurde diese als **VIRTUAL** deklariert. In diesem Zusammenhang muß eine Prozedur als **CONSTRUCTOR** definiert werden, die garantiert als erste aufgerufen wird und die Tabelle für die Zuordnung der virtuellen Methoden initialisiert. Diese Zuordnung braucht zum Zeitpunkt des Kompilierens noch nicht bekannt zu sein (auch späte Bindung genannt). Unterbleibt die Initialisierung, so ist ein Systemabsturz sicher. Zur Vereinfachung wurde die Syntax für den Aufruf der Standardprozedur *New* erweitert, die jetzt auch mit zwei Parametern aufgerufen werden kann. Damit verkürzen sich die beiden oben beschriebenen Anweisungen zu

New(liste,init);

Soweit erscheint diese Vorgehensweise nicht sensationell, aber wir können diese Liste jetzt als *Urahn* für alle weiteren Listen nutzen, das heißt, die entsprechenden Prozeduren müssen nicht wieder neu programmiert werden. Als Beispiel soll eine Liste von Buchtiteln alle Eigenschaften der alten Liste *erben* (und die betreffen die Verwaltung) und zusätzliche Datenfelder für den Titel, und die Bestellnummer erhalten – das Feld *Autor* dient hierbei als Schlüssel. Und so sieht der neue Typ aus:

```
type
buecher_typ = object (liste_typ)
  titel        : string;
  bestellnr    : string;
end;
```

Alle vererbten Methoden und Daten sind sofort nutzbar:

```
var buecher: ^buecher_typ;
New(buecher,init);
buecher^.key:='Raymond Chandler';
```

Natürlich können jederzeit weitere Listen definiert werden, etwa eine für englische Bücher, die zusätzlich den Originaltitel enthält:

```
unit listrout;

INTERFACE

type

listeptr = ^liste_typ;

liste_typ = object
 key        : string;
 next       : listeptr;
 constructor init;
 procedure add(liste,elem: listeptr);
 procedure delete(liste: listeptr);
 procedure show(liste: listeptr);
 destructor done;
end;

IMPLEMENTATION

constructor liste_typ.init;
begin
self.next:=NIL;
end;

procedure liste_typ.add;
begin
elem^.next:=liste^.next;
liste^.next:=elem;
end;

procedure liste_typ.delete;
var one_elem: listeptr;
begin
one_elem:=liste^.next^.next;
dispose(liste^.next,done);
liste^.next:=one_elem;
end;

procedure liste_typ.show;
var one_elem: listeptr;
begin
one_elem:=liste;
repeat
writeln(one_elem^.key);
one_elem:=one_elem^.next;
until one_elem=NIL;
end;

destructor liste_typ.done;
begin
end;

end.
```

```
program listest;

uses listrout;

type testptr = ^testtyp;
     testtyp = object (liste_typ)
               strasse    : string;
               stadt      : string;
               end;

var  liste,elem : testptr;

begin
New(liste,init);
liste^.key:='Pit';
New(elem,init);
elem^.key:='Roger';
liste^.add(liste,elem);
New(elem,init);
elem^.key:='Tucky';
liste^.add(liste,elem);

liste^.show(liste);

liste^.delete(liste);
dispose(liste,done);

end.
```

***Bild 1** Das Objekt Liste mit zugehörigen Methoden und kleinem Testprogramm*

type
books_typ = object (liste_typ)
  titel : string;
  orgtitel : string;
  bestellnr : string;
end;

Die entstandene Objekthierarchie hätte nun folgendes Aussehen:

```
      liste
      /  \
buecher  books
```

Da sich beide neuen Listen nur durch ein zugefügtes Feld unterscheiden, wäre es einfacher, die letzte Liste direkt aus der vorigen zu erzeugen, und zwar mit

books_typ = object (buecher_typ)
  orgtitel : string;
end;

In diesem Fall entsteht folgende Objekthierarchie:

```
 liste
   |
buecher
   |
 books
```

Es ist anzumerken, daß neue Objekte ihre Vorfahren stets vollständig enthalten; es können Daten und Methoden hinzugefügt oder umdefiniert (überlagert) werden – nie aber entfernt. In diesem Zusammenhang wurde auch die Zuweisungskompatibilität erweitert. In der Hierarchie höher stehenden Objekten können ihre Nachfahren zugewiesen werden (obwohl sie nicht typkompatibel sind), aber nicht umgekehrt. In Bild 1 wird das Objekt Liste einmal sehr einfach dargestellt. Eigene Ideen zur Erweiterung sind gefragt. Das zugehörige Testprogramm erweitert das Objekt mit einem Namen als Schlüssel um die Adresse (Stadt und Straße). Das Beispiel zeigt, wozu die Deklaration der Routine *show* als **VIRTUAL** dient. Jetzt wird nur der Name angezeigt; es wäre aber jederzeit möglich, eine erweiterte Prozedur gleichen Namens für das neue Objekt anzulegen, die an die neuen Aufgaben angepaßt ist (Anzeige der vollständigen Adresse).

Für den expliziten Zugriff auf Komponenten des eigenen Objekts kann der *Self*-Operator verwendet werden. Ist das Zielobjekt eindeutig bestimmt, so kann er auch weggelassen werden. Beim Zugriff auf Komponenten anderer Objekte müssen diese ähnlich wie bei Records qualifiziert werden.
Beispiele:
obj1.feld – Zugriff auf die Komponente *feld* von *obj1*
self.feld – Zugriff auf die Komponente *feld* des *eigenen* Objekts

Auf die Bedeutung der **CONSTRUCTOR**-Routine im Zusammenhang mit virtuellen Objekten wurde bereits hingewiesen; es kann sich hierbei um eine leere (Dummy-) Routine handeln. Zum Entfernen von Listenelementen aus dem Speicher muß eine weitere Routine als **DESTRUCTOR** definiert werden, die im Zusammenhang mit der Dispose-Funktion die Größe der Objekte bereitstellt, was aufgrund der virtuellen Methoden komplizierter ist, als bei Turbo 5.0 /1/.
Noch einmal Klartext: Eine **CONSTRUCTOR**-Routine muß stets bei Verwendung von **VIRTUAL**-Prozeduren vorgesehen werden – unabhängig davon, ob im Speicher dynamisch (mit Listen) gearbeitet wird oder nicht. Eine DESTRUCTOR-Routine dagegen ist nur nötig, wenn Objekte als dynamische Strukturen im Speicher freigegeben werden sollen.
Eine kleine Einschränkung beim Umdefinieren von Routinen gibt es allerdings: Der Kopf (einschließlich Parameter) muß identisch mit der ursprünglich deklarierten Routine sein. Ein Ausweg ist aber auch hier möglich: Der Definition einer zusätzlichen Routine mit geändertem Prozedurkopf steht nichts im Wege.
Die wirklichen Vorteile virtueller Methoden kommen im vorangegangenen Beispiel leider nicht ausreichend zur Geltung. Schauen wir uns deshalb einmal den folgenden Fall an: Eine Routine überträgt einen Block über die serielle Schnittstelle. Da es sich um Texte handelt, wird eine Prozedur verwendet, die eine Zeile sendet und wiederum eine Prozedur zum Senden eines Zeichens nutzt (Programmrahmen für unit *sendrout* in Bild 2). Das Programm *testalt* ruft diese Unit auf. Soll nun die Übertragung über eine andere Schnittstelle erfolgen, so wäre es ausreichend, die Zeichenübertragungsroutine zu ändern. Ist der Quelltext vorhanden, so kann die Routine dort ausgetauscht werden, wenn nicht, muß alles neu programmiert werden. Eine Deklaration als **VIRTUAL** ermöglicht die Neudefinition nur der innersten Routine ohne Neukompilation der unit *sendrout*. Damit werden grundsätzliche Prinzipien prozeduraler Programmierung außer Kraft gesetzt – nämlich die der lokalen Gültigkeit von Prozeduren und Funktionen!
Die wenigen Beispiele zeigen bereits die vielen neuen Möglichkeiten. Wenn bis jetzt nur von Prozeduren die Rede war, so sei hinzugefügt, daß selbstverständlich auch Funktionen in gleicher Weise genutzt werden können.
Besonders beeindruckend ist die Behandlung graphischer Objekte /1/. Es ist durchaus möglich, konsequent objektorientiert zu programmieren und auf Datenzugriffe weitestgehend zu verzichten. Statt einer Zuweisung der Art
buecher^.key:='Graham Greene';
muß dafür eine eigene Methode definiert werden:
procedure buecher.fill('Graham Greene', . . .);
Die Frage, inwieweit diese Methode sinnvoll ist und wann sie ausschließlich zum Programmoverhead führt, ist von jedem Programmierer selbst zu beantworten.

Zusammenfassung

Turbo-Pascal 5.5 ist, wie bereits erwähnt, vollständig kompatibel zur Version 5.0 und wurde um nur 4 Schlüsselworte erweitert:
OBJECT
VIRTUAL
CONSTRUCTOR
DESTRUCTOR
In Verbindung damit ist die Syntax der Standardprozeduren *New* und *Dispose* erweitert worden. Eine weitere Verbesserung betrifft die Overlaytechnik. Der Puffer ist jetzt als Ringpuffer organisiert, was zu einer Optimierung der Einlagerungszugriffe genutzt werden kann. Dafür gibt es zwei neue Standardprozeduren. Ferner ist jetzt ein Nachladen aus dem EXE-File möglich.
Der On-Line-Debugger präsentiert sich um objektorientierte Optionen erweitert und das Help-System ist weiter verfeinert worden. Als Stand-Alone-Debugger wird die neue Version 1.5 des Turbo-Debuggers geliefert (im Turbo-Assembler-Paket), mit dem beispielsweise der Objektbaum graphisch dargestellt werden kann (Bild 3).
Turbo-Pascal 5.5 beinhaltet die grundlegenden Ideen objektorientierter Programmierung: Kapselung, Vererbung und Polymorphie. Im Objekt-Typ, der dem Record-Typ nahe verwandt ist, werden Daten und Algorithmen (Methoden genannt) gemeinsam deklariert. Die Methoden sind also an die Daten gebunden. Für dieses Vorgehen steht der Begriff *Kapselung*. Wie bereits erwähnt, wird die Kapselung nicht erzwungen; ein direkter Datenzugriff ist weiterhin möglich.

```
unit sendrout; (* unit mit Senderoutinen *)

interface

type block = object
     buffer:string;
     constructor init;
     procedure send_buffer; virtual;
     procedure send_line; virtual;
     procedure send_char; virtual;
     end;

implementation

(* Dummyroutine, nur als Constructor *)
constructor block.init;
begin
end;

(* Soll ein Zeichen uebertragen, zum Test
   erfolgt nur eine Ausschrift *)

procedure block.send_char;
begin writeln('Alte Routine');
end;

procedure block.send_line;
begin
.
send_char;
.
end;

procedure block.send_buffer;
begin
.
send_line;
.
end;

end.
```

```
program sendalt;

uses sendrout;

var buffer: block;

begin
buffer.init;
buffer.send_buffer;
end.
```

```
program sendneu;
(* definiert eine eigene Routine
   und ruft diese auf           *)

uses sendrout;

(* Definieren eines neuen Objektes, dass die Routinen
   send_buffer und send_line erbt, waehrend send_char
   neu implementiert wird *)

type newblock = object (block)
                procedure send_char; virtual;
                end;

var buffer:newblock;

procedure newblock.send_char;
begin writeln('Neue Routine');
end;

begin
buffer.init;
buffer.send_buffer;
end.
```

Bild 2 Beispiel für die vorteilhafte Verwendung virtueller Methoden

```
 File  View  Run  Breakpoints  Data  Window  Options                 READY
┌Module: ODEMO  File: ODEMO.PAS 36──────────────────────────────────────────1┐

    IntNodePtr = ^IntNode;
    IntNode = object(TestNode)
      Value: Longint;
      constructor Init(V: Longint);
      constructor Load(var S: Stream);
      procedure Dump; virtual;
      procedure Store(var S: Stream);
    end;
           ┌Object Hierarchy═══════════════════════════════════════3┐
    Strin║BASE           │SPROC
         ║BUFSTREAM      │BASE
    StrNo║DOSSTREAM      │├──────STREAM
    StrNo║INTNODE        ││      └───────DOSSTREAM
      Val║LIST           ││              └───────BUFSTREAM
      con║NODE           ││                      └───────TESTSTR
      des║SPROC          │└──────NODE
      con║STREAM         │       └───────TESTNODE
─────────║STRNODE        │               ├───────INTNODE
┌Watches─║TESTLIST       │               └───────STRNODE             ──────2┐
         ║TESTNODE       │LIST
F1-Help F2-Bkpt F3-Close F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run
```

***Bild 3** Der Turbo-Debugger stellt den Objektbaum grafisch dar*

Die Beispiele haben gezeigt, wie auf verschiedenen Ebenen der Objekthierarchie gleichnamige Prozeduren unterschiedlich implementiert sein können, was als *Polymorphie* (griechisch: Vielgestaltigkeit) bezeichnet wird. Wichtig ist die Aussage, daß die Zuordnung zum Zeitpunkt des Compilerens nicht bekannt sein muß, was Änderungen an Software ermöglicht, die als Objektcode vorliegt. In Gegensatz zu C++ erlaubt Turbo-Pascal kein Überladen der Standardoperatoren.

Die *Vererbung* bewirkt, daß einmal programmierte Datenstrukturen und Unterprogramme auch in nachfolgend definierten Objekten Verwendung finden können – Zufügen und Umdefinieren eingeschlossen. Grundlage dafür ist die Einteilung der Objektstrukturen in Klassen. Jede Klasse enthält alle Eigenschaften übergeordneter Klassen als Gemeinsamkeiten; Unterschiede werden neu hinzugefügt. Es handelt sich hier um einen Vorgang, wie er bei jeder Einordnung neuer Elemente in bestehende Strukturen üblich ist.

Noch ein Wort zum ebenfalls jüngst erschienenen Quick-Pascal 1.0. Es ist vollständig kompatibel zu Turbo 5.0 und enthält vergleichbare objektorientierte Erweiterungen wie Turbo 5.5. Hier aber bestehen geringfügige syntaktische Differenzen.

Literatur

/1/ Heimsoeth Software GmbH & Co Produktions- und Vertriebs-KG: Handbuch Turbo-Pascal, Version 5.5, München 1989

/2/ Mittendorfer, J.: Smalltalk – Die Hintergründe der objektorientierten Programmierung. mc, München (1989) 3

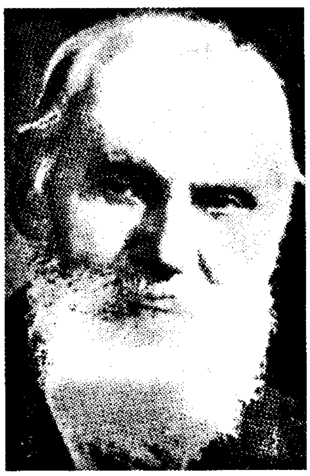
/3/ Dohmen, N.: Objekt mit Zukunft. mc, München (1989) 9

/4/ Stal, M.: Turbo-Pascal 5.5 und Quick-Pascal. CHIP, München (1989) 7

/5/ js/mti: Feiner Unterschied – Turbo Pascal und Quick-Pascal. PC Praxis (1989) 9

/6/ Zerbe, K.: Pascal Plus Plus. c't, Hannover (1989) 9

Wegbereiter der Informatik



SIR WILLIAM THOMSON (LORD KELVIN)

*** 1824 Belfast**
† 1907 Netherhall (Schottland)

Wie sah der erste Analogrechner aus? Wie war er konstruiert, wie funktionierte er? Existiert noch das Original oder ein Modell? Diese Fragen sind vorerst noch offen. So viel ist jedoch sicher: Zur Entwicklung der Analogrechentechnik hat der englische Physiker Sir William Thomson einen entscheidenden und wegbereitenden Beitrag geleistet. Und zwar hat er das grundlegende analogrechentechnische Programmierprinzip zur Lösung gewöhnlicher Differentialgleichungen angegeben, das ganz wesentlich auf dem Rückkopplungsprinzip beruht und bis heute verwendet wird.

In den Mitteilungen der Londoner Royal Society, den Proceedings vom Jahre 1876, schreibt Thomson, daß er viel Mühe darauf verwendet habe, einen mechanischen Integrierer zu konzipieren; er erwähnt dabei mehrmals einen „Platten-, Kugel- und Zylinderintegrierer" seines Bruders, dem vermutlich die technische Realisierung zu verdanken ist. Es folgen dann ausführliche Hinweise, wie für gewöhnliche lineare Differentialgleichungen 2. Ordnung das Rückkopplungsprinzip mit Hilfe zweier Integrierer verwirklicht werden kann, sowie Begründungen für die Rechtfertigung dieses Prinzips. Außerdem wird das Lösungsprinzip noch auf gewöhnliche lineare Differentialgleichungen höherer Ordnung mit variablen Koeffizienten übertragen und schließlich für nichtlineare Differentialgleichungen erweitert.

Studiert hat Thomson 1834–45 an den Universitäten Glasgow und Cambridge, danach arbeitete er ein Jahr in Paris bei dem Physiker und Chemiker Henri Victor Renault (1810–78). In den Jahren 1846–99 war er in Glasgow Professor für Theoretische Physik, 1851 wurde er Mitglied der Royal Society und war 1890–95 deren Präsident. Ab 1896 war er Ehrenmitglied der Petersburger Akademie. Für seine wissenschaftlichen Verdienste wurde er 1892 zum *Lord Kelvin of Largs* geadelt.

Schon als Student veröffentlichte Thomson einige Arbeiten über die Anwendung von Fourierreihen bei verschiedenen Problemen der Physik. Nahezu gleichzeitig mit R. J. E. Clausius (1822–88) formulierte er 1851 den *2. Hauptsatz der Thermodynamik*. Zusammen mit J. P. Joule (1818–89) untersuchte er 1853–54 die Temperaturänderung von Gasen bei Expansion ohne äußere Arbeitsleistung; dabei wurde der *Joule-Thomson-Effekt* gefunden, der die Grundlage der technischen Gasverflüssigung darstellt. Des weiteren erforschte Thomson die induktiven und kapazitiven Eigenschaften von Überseekabeln, für die er eine Theorie der elektromagnetischen Schwingungen ausarbeitete *(Thomsonsche Formel)*. Er befaßte sich auch mit verschiedenen Fragen der Hydrodynamik, der Geophysik und der Astrophysik. Von ihm stammt die Hypothese vom dereinst eintretenden Wärmetod des Weltalls.

Thomson stellte eine wissenschaftlich begründete Definition der (heute in Kelvingraden angegebenen) absoluten Temperatur auf und konstruierte verschiedene physikalische Meßgeräte, zum Beispiel das Quadranten-Elektrometer. Er verbesserte auch das Spiegelgalvanometer und den Magnetkompaß.

Dr. Klaus Biener

TERMINE

V. Informatik-Kolloquium „Computergrafik in CAE-Systemen"

WER? Technische Hochschule „Carl Schorlemmer" Merseburg
WANN? 29. März 1990
WO? Merseburg
WAS?
- Einsatz der grafischen und der Geometriedatenverarbeitung im Ingenieurwesen
- Integration der Computergrafik in CAE-Systemen

WIE? Interessenten wenden sich bitte an die Technische Hochschule „Carl Schorlemmer" Merseburg, Rechenzentrum, Otto-Nuschke-Straße, Merseburg, 4200; Tel. 46 29 87 oder 46 29 62.

Dr. Reinemann

Anwendersymposium Rechnernetze

WER? Wissenschaftlich-Technische Gesellschaft für Meß- und Automatisierungstechnik und VEB Robotron-Vertrieb Berlin
WANN? 6. bis 7. März 1990
WO? Rathaus der Stadt Dresden, Festsaal
WAS?
- Gateways (ohne ESER-LANs)
- ESER-LANs
- Bürodienste Rolanet 1
- Datennetz der Deutschen Post Post/WANs
- Einsatz von Lichtwellenleitern im Rolanet

WIE? Teilnahmemeldungen bitte schriftlich an: Präsidium der KDT, WGMA, PSF 1315, Berlin, 1086; Tel. 2 26 53 04, 2 26 52 18

Müller

Einführung in Forth-83

Teil 6

Dr. Hartmut Pfüller (Leiter),
Dr. Wolfgang Drewelow,
Dr. Bernhard Lampe, Ralf Neuthe,
Egmont Woitzel, Rostock

6. Das Forth-Dialogsystem

6.1. Die Struktur des Dialogsystems

Zum Dialogsystem zählt man alle diejenigen Worte, die – grob gesagt – zum Komplex der Ablaufsteuerung für die Kommunikation gehören. Das sind der Kaltstart, der Textinterpreter und das Fehlersystem.

6.1.1. Initialisierung

Beim Start eines Forth-Systems muß zuerst dessen Initialisierung vorgenommen werden. Diese Aufgabe übernimmt in der Regel das Wort **COLD**. Dessen Anwesenheit ist im Standard Forth-83 zwar nicht vorgeschrieben, es ist aber in fast jedem System vorhanden. COLD ist der Hochwort-Eintrittspunkt für den Systemstart. Im wesentlichen werden damit die Speicheraufteilung festgelegt, das System nach den Festlegungen aus dem Boot-Bereich voreingestellt, die Stapel initialisiert, eine Systemausschrift zum Ausgabegerät gesendet und schließlich die Steuerung an den Textinterpreter übergeben.

6.1.2. Dialogführung

Der Dialog mit dem Bediener erfolgt unter Steuerung des Textinterpreters **QUIT**. Der Standard verlangt, daß QUIT den Rückkehrstapel säubert, den Ausführmodus einschaltet, die Eingabebereitschaft über das aktuelle Eingabegerät herstellt und die Wortinterpretation durchführt.

Innerhalb dieser Richtlinien kann der Rahmenablauf weitgehend variiert werden. Eine mögliche Variante ist im Bild 6.1 dargestellt. Hier wird QUIT mit der **DOER-MAKE**-Konstruktion /1/ vektoriert ausgeführt. Damit ist der gesamte Textinterpreter auch für Fachsprachen, die für Forth untypisch sind, komplett austauschbar.

6.1.3. Der Eingabestrom

Der Eingabestrom ist die Zeichenfolge, die vom Textinterpreter des Systems verarbeitet wird. Er kann sowohl vom aktuellen Eingabegerät (über den Texteingabepuffer) als auch vom Massenspeicher (über einen Blockpuffer) geholt werden. **BLK >IN TIB #TIB** spezifizieren den Eingabestrom. Worte, die auf BLK >IN TIB #TIB zugreifen oder deren Inhalte verändern, sind verantwortlich für das Aufrechterhalten (Sichern) und das Rückspeichern der Informationen zur Steuerung des Eingabestromes /2/.

Woher der Eingabestrom zu entnehmen ist, wird durch die Variable **BLK** spezifiziert. Sie enthält die Nummer des entsprechenden Massenspeicherblockes oder eine Null, wenn der Eingabestrom aus dem Texteingabepuffer kommen soll. Die Anfangsadresse dieses Texteingabepuffers kann mit **TIB** (Terminal-Input-Buffer) geholt werden. In der Variablen **#TIB** wird die aktuelle Anzahl der dort vorhandenen gültigen Bytes gehalten.

Die sequentielle Entnahme von Zeichenketten wird durch die Variable >**IN** unterstützt. Sie enthält den aktuellen Zeichenoffset zu der Stelle, an der die nächste Entnahme (aus dem Block- oder dem Texteingabepuffer) erfolgen kann. Bild 6.2 zeigt das Zusammenspiel der Worte.

WORD ist das zentrale Wort für die zerstörungsfreie Entnahme von Teilzeichenketten (typischerweise Namen oder Zahlen) aus dem aktuellen Eingabestrom. Dafür verwendet WORD ein Begrenzerzeichen, das ihm als Parameter auf dem Stapel übergeben wird. Vor der Kette befindliche Begrenzerzeichen werden übergangen, das entspricht einem Herantasten an den interessierenden Teil. Danach werden so lange Zeichen entnommen, wie das Begrenzerzeichen nicht angetroffen wird. WORD stellt dann die Variable >IN auf die Stelle im Eingabestrom ein, die unmittelbar hinter dem abschließenden Begrenzerzeichen liegt. So ist mit einer erneuten Ausführung von WORD die nächste Kette erreichbar. Ist der Eingabestrom erschöpft, liefert WORD eine leere Kette (Zählbyte = 0). WORD übergibt als Ergebnisparameter eine Adresse, die anzeigt, wo die separierte Kette im Standardformat abholbereit liegt.

6.1.4. Fehlerbehandlung

Das Fehlersystem von Forth achtet vor allem auf fatale Fehler wie Stapelunterlauf, verkehrte Verschachtelung von Steuerstrukturen, Wörterbuchüberlauf usw. Es schützt nicht vor Systemabstürzen infolge von liegengelassenen Parametern auf dem Rückkehrstapel oder nicht autorisiertem Überschreiben von Programmteilen.

Tests auf bestimmte Fehlerbedingungen können zu unterschiedlichen Zeitpunkten und in verschiedenen Worten stattfinden. Compilerworte wie **IF**, **ELSE** und **THEN** achten beispielsweise in vielen Systemen eigenständig auf ihre richtige Kombination.

Dem Anwender stellt Forth das Wort **ABORT"** zur Verfügung. Es wird in der Form

Flag ABORT" Fehlertext"

verwendet. Ist das Flag falsch, wird mit der Bearbeitung hinter ABORT"..." fortgesetzt. Ansonsten erfolgt die Ausgabe des Fehlertextes und der Eintritt in die Fehlerroutine ABORT.

6.2. Beispiel: Universeller Server und lernender Klient

Mit einem etwas größeren Programmbeispiel soll der Kurs Forth-83 nun beendet werden. Ausgewählt wurde hierfür eine Rechnerkopplung (A 5120 und A 7150) über IFFS-Leitung, wobei Kopplungen BC – BC, BC – AC oder AC – AC möglich sind. Obwohl das für diesen Kurs als Prinzipdemonstration entwickelte Programm in vieler Hinsicht perfektioniert werden kann, haben Versuche gezeigt, daß es auch in dieser Version mit praktischem Nutzen einsetzbar ist.

6.2.1. Ausgangspunkt

In Rechnernetzen ist es üblich, daß ein oder mehrere Rechner als sogenannte Server (etwa soviel wie Diener) fungieren. Ihre Aufgabe ist die Ausführung bestimmter Dienste, die über das Netz angefordert werden können (z. B. Recherchen in Datenbanken). Zu diesem Zweck muß ein Rechner über die Kommunikationsverbindung Aktionen auf einem anderen Rechner auslösen können. Das hier vorgestellte Programm zeigt, wie solche Dienste via Schnittstelle abgefordert werden können.

6.2.2. Das Konzept

Im Beispiel werden zwei Rechner bidirektional miteinander verbunden. Einer erhält das Rahmenprogramm für den Server und der andere das für den Klienten.

Zuerst muß auf dem Server der Loginprozeß

```
MAKE QUIT      ( QUIT ist ein DOER, der folgendes machen soll: )
>SYSTEM        ( Einschalten des Systemmodus                    )
CHECK          ( testen, ob alles in Ordnung ist; ggf. Ordnung )
               ( herstellen [z. B. Stapelüberlauf korrigieren] )
[COMPILE] [    ( Ausführmodus einschalten                       )
BEGIN          ( Beginn der Endlosschleife des Textinterpreters)
  RP0 RP!      ( Initialisierung des Rückkehrstapels           )
  PROMPT       ( Eingabebereitschaft signalisieren             )
  QUERY        ( Entgegennahme der Eingaben                    )
  >PROCESS     ( Nutzermodus aktivieren                        )
  INTERPRET    ( eigentliche Interpretation des empfangenen    )
               ( Textes; Ausführung/Kompilation und Fehler-    )
               ( test, ggf. Ausstieg zum Fehlersystem          )
  >SYSTEM      ( Umschaltung auf Systemmodus                   )
  ANSWER       ( Antwort, daß kein Fehler bemerkt wurde        )
AGAIN          ( unbedingt zurück zum Anfang der Schleife      )
;              ( dieses Semikolon wird deshalb nie erreicht    )
```

Bild 6.1 Kodierung von QUIT bei comFORTH 2

```
BLK    !        =0          !        <>0
       ! Entnahme des Eingabestroms erfolgt aus dem
       ! Texteingabepuffer !     Blockpuffer
-------+-------------------+-------------------------
adr    !        TIB        ! Blockpufferadresse
#byte  !        #TIB       ! 1024=Blocklänge


-----------------------------------------------------
 !           !                            !
 !entnommen  !nächster  übernächster      !ungültig
 !           !                            !
-----------------------------------------------------
 ^           ^                            ^
 !           !                            !
adr        adr+>IN                     adr+#byte
```

Bild 6.2 Herkunft des Eingabestroms (oben) und Zählweise der entnommenen Zeichen (unten)

(**LOGIN** auf der Datei SERVER.CF2) gestartet werden. Das gibt dem Klienten die Möglichkeit zur Aufnahme der Verbindung (ebenfalls mit LOGIN; Datei CLIENT.CF2). In unserem Beispiel erwartet der Server als erste Mitteilung ein Byte, das als Kennungsnummer des jeweiligen Klienten angesehen wird. Nach dessen Eintreffen wird zuerst die Codefeldadresse des Wortes **SFIND** und dann ein Quittungsbyte „Null" an den Klienten übermittelt; anschließend wird der Serverbetrieb durch **SERVE** aufgenommen. Von nun an kann der Klient den Server zu beliebigen Aktionen (Ausführung von Forthworten) anregen. Dafür findet die folgende Methode Verwendung: Der Klient wartet aktiv auf die Quittung „Null" vom Server und beginnt dann mit dem Senden von 16 Bit. Der Server befindet sich im Wort SERVE, wo die entgegengenommene Zahl als eine Codefeldadresse angesehen und das entsprechende Wort per **EXECUTE** ausgeführt wird. Anschließend wird wieder eine Null als Quittung an den Klienten zurückgesendet, die diesem die Bereitschaft für neue Serverdienste signalisiert.
Der Klient ist in der Lage, noch unbekannte Codefeldadressen in Erfahrung zu bringen, er kann so lernen, Dienst für Dienst rationeller als über den Namen in Anspruch zu nehmen. Dazu löst er zuerst die Ausführung des Serverwortes SFIND (siehe Datei CLIENT.CF2) aus. Danach wird mit **PUTS** der Name des zu suchenden Wortes als Zeichenkette übermittelt, die der Server mit **GETS** abholt und mit FIND in seinem Wörterbuch sucht. Als Ergebnis der Suche übergibt der Server zwei Zahlen mittels **PUT PUT** an den Klienten, der sie mit **GET GET** entgegennimmt. Ob das gesuchte Wort im Server gefunden werden konnte, zeigt im Klienten die Zahl auf dem Stapel als Flag an. Ist es „wahr", liegt auf dem SOS (second of stack) die dazugehörige Codefeldadresse, andernfalls ein uninteressanter Wert.
Damit der Bediener mit dem hier erläuterten Mechanismus nicht belastet wird, steht ihm das Definitionswort **SERVICE:** für die bequeme Inanspruchnahme von dem Namen nach bekannten Serverdiensten zur Verfügung. Es wird wir folgt benutzt:

SERVICE: Dienstname Serverwort

Nach der Definition von „Dienstname" kann durch dessen Aufruf die Ausführung des dazugehörigen „Serverwortes" veranlaßt werden. Bei der Definition von „Dienstname" wird im Klienten ein neuer Wörterbucheintrag eingerichtet. In dessen Parameterfeld befinden sich 2 Byte für ein Feld zur Aufnahme in die Liste aller Servicedienste (Ankerfeld **SERVICE^**), 2 Byte (mit Null initialisiert) zum Abspeichern der vom Server übermittelten Codefeldadresse und einige weitere Bytes für den Namen des korrespondierenden Serverwortes als Zeichenkette. Bei Aufruf von „Dienstname" (**DOES>**-Teil von SERVICE:) wird zuerst nachgeschaut, ob sich in der Zelle für die Codefeldadresse bereits ein von Null verschiedener Wert befindet. In diesem Fall kann sofort die Ausführung des Dienstes veranlaßt werden; andernfalls wird die Codefeldadresse erst mit SFIND in Erfahrung gebracht.

Das Wort **LOGOFF** ermöglicht ein Abmelden des Klienten vom Server. Dazu werden auf dem Server der Loginprozeß neu gestartet und anschließend die Codefeldadressen aller Serverdienste im Klienten mit Null überschrieben. So ist ein Wechsel des Servers möglich, ohne daß Konflikte aufgrund verschiedener Codefeldadressen für die einzelnen Dienste auftreten.

6.2.3. Zum Programmaufbau
Die Software beider Kommunikationspartner besteht im wesentlichen aus drei aufeinander aufbauenden Schichten. In der untersten Schicht wird die physische Bedienung der Schnittstelle ermöglicht. Dazu gehören die Funktionen **TX** „Byte senden", **TX?** „Senderstatus", **RX** „Byte empfangen", **RX?** „Empfängerstatus" und **0!X** „Schnittstelle initialisieren". Sie sind auf den Dateien IFSS5120.CF2 für den Bürocomputer A 5120 und IFSS 7150.CF2 für den Arbeitsplatzcomputer A 7150 codiert. Eine besondere Aufgabe kommt der physischen Initialisierung zu. Durch sie müssen (noch vor dem logischen LOGIN) nicht nur die Schnittstellen auf den gewünschten Sende- und Empfangsmodus eingestellt, sondern auch das „Einrasten" vollzogen werden. Das ist notwendig, da nach der Initialisierung der Peripheriebausteine die Sendepuffer nicht in jedem Fall leer sind.
In der zweiten Schicht (Datei TRANSFER.CF2) wird das überwachte Senden und Empfangen von Bytes (**CPUT** und **CGET**), von Worten (PUT und GET) und von Zeichenketten (PUTS und GETS) durchgeführt. Nur in der obersten Schicht unterscheiden sich die Programme von Server (SERVER!CF2) und Klient (CLIENT.CF2).

6.2.4. Ein Sitzungsbeispiel
Zuerst ist auf beiden Rechnern die Schnittstelle physisch mit **ALSO COM 0!X** zu initialisieren. Danach muß auf dem Server durch Aufruf des Wortes LOGIN der Loginprozeß gestartet werden. Damit ist der Server für den Klienten verfügbar. Von nun an arbeitet der Server ferngesteuert, und vom Klienten aus kann die Verbindung mit

12 LOGIN*(cr) ok*

aufgebaut werden, wobei die 12 die Kennung des Klienten darstellen soll.

Im folgenden Benutzungsbeispiel ist es das Ziel, einen Block vom Massenspeicher des Servers zu lesen und zum Klienten zu übertragen. Das erfordert erst noch die Vereinbarung eines Dienstes für die Aktion „Block lesen":

SERVICE: SBLOCK BLOCK*(cr) ok*

Nach der Übermittlung einer Blocknummer (hier 33) kann der Klient den Dienst **SBLOCK** veranlassen:

33 >S SBLOCK*(cr) ok*

Zur Übertragung des Speicherblocks zum Klienten sind noch die Parameter für **SPUTS** (PUTS auf dem Server) bereitzustellen. Die Anfangsadresse des zu übertragenden Datenbereichs von BLOCK liegt dort noch auf dem Stapel. Die Anzahl der zu sendenden Bytes entspricht der Länge eines Massenspeicherblocks (gleich 1024) und muß mit **>S** zum Stapel des Servers transferiert werden, bevor die Übertragung beginnen kann:

1024 >S SPUTS*(cr) ok*

Mit GETS nimmt nun der Klient die Daten in Empfang:

PAD 1024 GETS DROP*(cr) ok*

In diesem Beispiel wurden die Daten nach **PAD** gebracht. Die ganze Aktion ließe sich natürlich auch als Programm in einem Forthwort definieren:

```
: BLOCK> (ps: n ===>) (Block empfangen)
  >S SBLOCK 1024 >S SPUTS
  PAD 1024 GETS DROP ;
```

Der Klient kann nun auf bequeme Weise beliebige Blöcke vom Server holen:

33 BLOCK>*(cr) ok*

Schluß

Literatur

/1/ Brodie, L.: In Forth denken. München: Carl Hanser Verlag 1986

/2/ Vack, G.-U.: Der Standard FORTH-83. Suhl: Kammer der Technik 1987

KONTAKT

Wilhelm-Pieck-Universität Rostock, Sektion Technische Elektronik, Albert-Einstein-Straße 2, Rostock, 2500; Tel. 40 52 14

Tafel 6 Kurzbeschreibung der Forthworte

Name	Stapeleffekt	Beschreibung
COLD	(===>)	Kaltstart, Initialisierung des Forthsystems
QUIT	(===>)	Ausführungsmodus, Eingabebereitschaft, Wortinterpretation beginnen
#TIB	(===> addr)	Variable, enthält Anzahl der Bytes im Texteingabepuffer
TIB	(===> addr)	Adresse des Texteingabepuffers, wo Zeichen vom Eingabestrom gepuffert sind
BLK	(===>)	Variable, enthält Nummer des Massenspeicherblocks oder 0 für Tastatur
>IN	(===>)	Variable, enthält aktuellen Zeichenoffset im Eingabestrom
WORD	(c===> addr)	anhand Trennzeichen c Teilzeichenkette aus Eingabe isolieren und nach addr schaffen
ABORT	(===>)	Parameterstapel säubern und Ausführen von QUIT
ABORT"	(?===>)	Bei Ausführung von ABORT" text" wird bei ?=wahr text ausgegeben, dann ABORT.

```
Datei A:CLIENT.LDR    Block 0
 0 CLIENT                                          Stand: EW 25-Oct-89
 1 ================================================================
 2                    Lader für den Klienten
 3 ================================================================
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
```

```
Datei A:SERVER.LDR    Block 1
 0 \ Ladeblock                                          EW 25-Oct-89
 1
 2  ONLY FORTH DEFINITIONS  DECIMAL
 3
 4 CR CR   .( -------- Lader fuer den Server )
 5
 6       @:ASM86      1 INCLUDE   \ 8086-Assembler
 7       @:IFSS7150   1 INCLUDE   \ Schnittstellenbedienung
 8       @:TRANSFER   1 INCLUDE   \ Transferdienste
 9       @:SERVER     1 INCLUDE   \ Serverinterface
10
11 CR       .( -------- fertig )
12 CR
13
14
15
```

```
Datei A:CLIENT.CF2    Block 1
 0 \ Rahmenprotokoll Klient                           RN 30-Oct-89
 1
 2  ONLY COM ALSO FORTH DEFINITIONS  DECIMAL
 3
 4 CR .( Warte ...lade Rahmenprotokoll Klient )
 5
 6    2 5 THRU
 7
 8 CR .(          ...fertig )
 9 CR
10
11  ONLY FORTH DEFINITIONS
12
13
14
15
```

```
Datei A:CLIENT.CF2    Block 2
 0 \     'SFIND LOGIN SEXECUTE SFIND                  EW 25-Oct-89
 1
 2 VARIABLE 'SFIND  ( cfa<SFIND> des Servers ) 'SFIND OFF
 3
 4 : LOGIN     ( ps: 8b ===> )( Klient 8b einloggen )
 5             ( Protokoll: t:8b r:addr )
 6             CPUT GET 'SFIND ! ;
 7
 8 : SEXECUTE  ( ps: addr ===> )( auf Server ausfuehren )
 9             ( Protokoll: r:8b t:addr )( letzter Fehler! )
10             CGET ?DUP IF ." Serverfehler " . QUIT THEN PUT ;
11
12 : SFIND     ( ps: addr ===> addr ? )( im Server suchen )
13             ( Protokoll: ..sexecute.. t:s r:addr r:? )
14             'SFIND @ ?DUP 0= ABORT" Server nicht geloggt"
15             SEXECUTE COUNT PUTS GET GET ;
```

```
Datei A:CLIENT.CF2    Block 3
 0 \     SERVICE^ SERVICE:                            RN 30-Oct-89
 1
 2 VARIABLE SERVICE^ ( Anker Serviceliste )  SERVICE^ OFF
 3
 4 : SERVICE: ( ps: ===> )( ib: local-name remote-name )
 5            ( definiert Serverdienst local-name, im )
 6            ( Server wird remote-name ausgefuehrt )
 7            CREATE HERE  SERVICE^ DUP @ ,  !  0 ,
 8                  BL WORD C@ 1+ ALLOT DOES>
 9            ( ps: ===> )( Aufruf von remote-name in Server )
10            DUP 2+ @ ?DUP               \ 'cf schon bekannt?
11            IF   SWAP DROP
12            ELSE DUP 4 + SFIND 0=
13                 ABORT" unbekannter Dienst" DUP ROT 2+ !
14            THEN SEXECUTE ;
15
```

```
Datei A:CLIENT.CF2    Block 4
 0 \     Standard-Dienstleistungen                    RN 30-Oct-89
 1
 2 SERVICE: SCPUT  CPUT          SERVICE: SCGET  CGET
 3 SERVICE:  SPUT   PUT          SERVICE:  SGET   GET
 4 SERVICE:  SPUTS  PUTS         SERVICE:  SGETS  GETS
 5 SERVICE: SLOGIN LOGIN
 6
 7 : >S ( ps: 16b ===> )( Transfer zum Server )
 8    ( Protokoll: ..sexecute.. t:16b )
 9    SGET  \ Abholen von 16 Bit durch den Server veranlassen
10    PUT ; \ 16 Bit (vom Klientenstapel zum Serverstapel) senden
11
12 : S> ( ps: ===> 16b )( Transfer vom Server )
13    ( Protokoll: ..sexecute.. r:16b )
14    SPUT  \ Senden von 16 Bit durch den Server veranlassen
15    GET ; \ 16 Bit (vom Serverstapel zum Klientenstapel) empf.
```

```
Datei A:CLIENT.CF2    Block 5
 0 \     LOGOFF                                       RN 30-Oct-89
 1
 2 : LOGOFF ( ps: ===> )( Abmelden beim Server )
 3          ( Protokoll: ..sexecute.. ..login.. )
 4          SLOGIN  0 'SFIND !          \ Server-Restart
 5          SERVICE^ BEGIN  @ ?DUP      \ Listenende?
 6                   WHILE  0 OVER 2+ ! \ 'cf-Feld ruecksetzen
 7                   REPEAT ;
 8
 9
10
11
12
13
14
15
```

```
Datei A:IFSS7150.CF2  Block 1
 0 \ Schnittstellenbedienung fuer A 7150              RN 30-Oct-89
 1
 2  ONLY FORTH DEFINITIONS  DECIMAL
 3
 4 CR .( Warte ...lade physische Schnittstellenbedienung )
 5
 6   2 5 THRU
 7
 8 CR .(          ...fertig )
 9 CR
10
11
12
13
14
15
```

```
Datei A:IFSS7150.CF2  Block 2
 0 \     COM AWAIT O!X                                EW 25-Oct-89
 1
 2 ALSO ROOT DEFINITIONS   VOCABULARY COM
 3      COM  DEFINITIONS
 4
 5 CODE COM ( ps: ax dx ===> ax )( COM-Interrupt ansprechen )
 6          DX AX MOV, BX DX MOV,
 7          20 # INT,  \ Interrupt 14H - COM-Schnittstelle BIOS
 8          AX BX MOV, DX POP, NEXT,
 9          END-CODE
10
11 : IFSS    ( ps: ax ===> ax )( Kanal COM2 )
12          1 COM ;
13
14
15
```

```
Datei A:IFSS7150.CF2  Block 3
 0 \     SUBFUNCTION (O!IFSS) (>IFSS) (IFSS>) IFSS? ...EW 25-Oct-89
 1
 2 : SUBFUNCTION ( ps: 8b ===> )( ib: name )( Unterfunktion def.)
 3               CREATE C, DOES>
 4               ( ps: al ===> ax )( Funktion 8b an IFSS )
 5               C@ JOIN IFSS ;
 6
 7 0 SUBFUNCTION (O!IFSS)
 8 1 SUBFUNCTION (>IFSS)
 9 2 SUBFUNCTION (IFSS>)
10 3 SUBFUNCTION IFSS?
11
12 : RX      ( ps: ===> 8b )( Zeichen lesen )
13           0 (IFSS>) 255 AND ;
14 : TX      ( ps: 8b ===> )( Zeichen schreiben )
15           (>IFSS) DROP ;
```

```
Datei A:IFSS7150.CF2  Block 4
 0 \     CTRL %RX? %TX? O!IFSS RX? TX?                HP 24-Oct-89
 1
 2 2 BASE !
 3 11101011 CONSTANT CTRL ( Steuerwort: 8 Bit Daten, 1 Stop )
 4                        ( mit ungeradem Paritaetsbit )
 5 0000000100000000 CONSTANT %RX? ( Empfangsdaten )
 6 0010000000000000 CONSTANT %TX? ( Sender leer )
 7
 8 DECIMAL
 9 : O!IFSS ( ps: ===> )( Initialisierung )
10          CTRL (O!IFSS) DROP ;
11 : RX? ( ps: ===> ? )( ungleich Null bei Empfangsdaten vorh. )
12       0 IFSS? %RX? AND ;
13 : TX? ( ps: ===> ? )( ungleich Null bei Sender leer )
14       0 IFSS? %TX? AND ;
15
```

```
Datei A:IFSS7150.CF2  Block 5
 0 \ Initialisieren der Serverschnittstelle           HP 24-Oct-89
 1
 2 HEX
 3
 4 : O!X    ( ps: ===> )( Initialisierung der Schnittstelle )
 5          ( zusammen mit Einrasten )
 6          O!IFSS 5 0 DO 0 TX LOOP 55 TX ;
 7
 8 DECIMAL
 9
10
11
12
13
14
15
```

```
Datei A:SERVER.LDR    Block 0
 0 SERVER                                             Stand: EW 25-Oct-89
 1 ================================================================
 2                       Lader für den Server
 3 ================================================================
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
```

```
Datei A:CLIENT.LDR    Block 1
 0 \ Ladeblock                                                EW 25-Oct-89
 1
 2  ONLY FORTH DEFINITIONS  DECIMAL
 3
 4 CR CR   .( -------- Lader fuer den Klienten )
 5
 6       @:IFSS5120   1 INCLUDE    \ Schnittstellenbedienung
 7       @:TRANSFER   1 INCLUDE    \ Transferdienste
 8       @:CLIENT     1 INCLUDE    \ Rahmenprotokoll Klient
 9
10 CR      .( -------- fertig )
11 CR
12
13
14
15
```

```
Datei A:IFSS5120.CF2  Block 1
 0 \ Schnittstellenbedienung fuer A 5120                     RN 30-Oct-89
 1
 2  ONLY FORTH DEFINITIONS  HEX
 3
 4 CR .( Warte ...lade physische Schnittstellenbedienung )
 5
 6    2 4 THRU
 7
 8 CR .(         ...fertig )
 9 CR
10
11  ONLY FORTH DEFINITIONS  DECIMAL
12 \S ------------------------------------------------------------
13 SIO auf der ASS des BC 5120, DFUe-Kanal: Tornummern 52H und 53H
14 Am Koppelbus ist eine Wickelverbindung vom CTC der ZRE zur SIO
15 der ASS erforderlich.
```

```
Datei A:IFSS5120.CF2  Block 2
 0 \ COM Tornummern & Initialisieren PIO, CTC                RN 30-Oct-89
 1 \ kleines Glossar fuer die Bedienung der Tore:
 2 \ CP!   ( ps: 8b port ===> )  Torausgabe von 8b auf "port"
 3 \ CP@   ( ps: port ===> 8b )  8b lesen vom Tor "port"
 4
 5 ALSO ROOT DEFINITIONS   VOCABULARY COM
 6      COM  DEFINITIONS
 7
 8 52 CONSTANT SIO     ( Datentor der SIO )
 9 53 CONSTANT PSIO    ( Programmier- & Statustor der SIO )
10 OC CONSTANT CTC     ( Tor Kanal 0 des CTC auf ZRE K 2526 )
11
12 : SIOINIT    ( ps: ===> )
13   OEA 05 OC1 3 0 1 45 4 18 ( Reset)  9 0 DO PSIO CP! LOOP ;
14
15 : CTCINIT    ( ps: ===> )               17 CTC CP!   1 CTC CP! ;
```

```
Datei A:IFSS5120.CF2  Block 3
 0 \ RX? TX? RX TX                                          HP 24-Oct-89
 1
 2 : RX?    ( -- ?)( wahr, falls Zeichen eingegangen & abholbereit)
 3   PSIO CP@ 1 AND ;
 4
 5 : TX?    ( -- ?)( wahr, falls Sender frei & senden moeglich)
 6   PSIO CP@ 4 AND ;
 7
 8 : RX     ( -- c)( eingegangenes Zeichen aus Empfangspuffer lesen)
 9   SIO CP@ ;
10
11 : TX     ( c --)( Zeichen zum Senden in Sendepuffer schreiben)
12   SIO CP! ;
13
14
15
```

```
Datei A:IFSS5120.CF2  Block 4
 0 \ Einrasten der Schnittstelle                             RN 30-Oct-89
 1
 2 : O!X    ( ps: ===> )( Initialisierung der Schnittstelle )
 3          ( zusammen mit Einrasten )
 4   CTCINIT SIOINIT
 5   BEGIN BEGIN RX? UNTIL RX   O= UNTIL
 6   BEGIN BEGIN RX? UNTIL RX 55 = UNTIL ;
 7
 8
 9
10
11
12
13
14
15
```

```
Datei A:SERVER.CF2    Block 1
 0 \ Rahmenprotokoll Serverinterface                         RN 30-Oct-89
 1
 2  ONLY COM ALSO FORTH DEFINITIONS  DECIMAL
 3
 4 CR .( Warte ...lade Rahmenprotokoll Server )
 5
 6    2 3 THRU
 7
 8 CR .(         ...fertig )
 9 CR
10
11  ONLY FORTH DEFINITIONS
12
13
14
15
```

```
Datei A:SERVER.CF2    Block 2
 0 \    ERROR? SERVE                                         RN 30-Oct-89
 1
 2 : ERROR?  ( ps: ===> 8b )( ermittelt Fehlernummern )
 3   0 ;     ( erster Ansatz: Eigentest immer in Ordnung )
 4
 5 : SERVE ( ps: ===> ... )( Dienstleistungen ueber Interface )
 6         ( Protokoll: r:addr ... t:8b )
 7         RPO RP!              \ Rueckkehrstapel initialisieren
 8         BEGIN  GET EXECUTE   \ Dienstleistung annehmen
 9                ERROR? CPUT   \ Eigentest und Quittung senden
10         AGAIN  ;
11
12
13
14
15
```

```
Datei A:SERVER.CF2    Block 3
 0 \    SFIND CLIENT# LOGIN                                  RN 30-Oct-89
 1
 2 : SFIND   ( ps: ===> )( 'cf-Suche ueber Interface )
 3           ( Protokoll: r:s t:addr t:? )
 4           PAD 1+ DUP 32 BLANK       \ PAD mit Leerzeichen fuellen
 5           31 GETS PAD C!            \ Wortname zum PAD empfangen
 6           PAD FIND SWAP PUT PUT ; \ Wort suchen; Ergebnis senden
 7
 8 VARIABLE CLIENT#
 9 : LOGIN   ( ps: ===> )( Verbindung mit Klienten aufbauen )
10         ( Protokoll: r:8b t:addr t:8b ..serve.. )
11         CGET CLIENT# C!     \ Nr. des Klienten holen & merken
12         ['] SFIND PUT       \ 'cf von SFIND senden
13         0 CPUT              \ Blindquittung senden
14         SERVE ;             \ Serverbetrieb aufnehmen
15
```

```
Datei A:TRANSFER.CF2  Block 1
 0 \ Transferdienste                                         RN 30-Oct-89
 1
 2  ONLY FORTH ALSO COM DEFINITIONS  DECIMAL
 3
 4 CR .( Warte ...lade Transferfunktionen )
 5
 6    2 3 THRU
 7
 8 CR .(         ...fertig )
 9 CR
10
11  ONLY FORTH DEFINITIONS
12
13
14
15
```

```
Datei A:TRANSFER.CF2  Block 2
 0 \     (CPUT) (CGET) CPUT CGET PUT GET                     RN 30-Oct-89
 1 \ SPLIT   ( ps: 16b ===> 8b-low 8b-high )  16b in 2x 8b zerlegen
 2 \ JOIN    ( ps: 8b-low 8b-high ===> 16b )  2x 8b zusammenfuegen
 3
 4 : (CPUT) ( ps: 8b ===> )( synchronisiertes Senden )
 5          BEGIN  TX?  UNTIL  TX ;
 6 : (CGET) ( ps: ===> 8b )( synchronisierter Empfang )
 7          BEGIN  RX?  UNTIL  RX ;
 8 : CPUT   ( ps: 8b ===> )( ueberwachtes Senden )
 9          DUP (CPUT) (CGET) - ABORT" Uebertragungsfehler" ;
10 : CGET   ( ps: ===> 8b )( ueberwachter Empfang )
11          (CGET) DUP (CPUT) ;
12 :  PUT   ( ps: 16b ===> )( Wort senden )
13          SPLIT SWAP CPUT CPUT ;
14 :  GET   ( ps: ===> 16b )( Wort empfangen )
15          CGET CGET JOIN ;
```

```
Datei A:TRANSFER.CF2  Block 3
 0 \     PUTS GETS                                           RN 31-Oct-89
 1
 2 : PUTS ( ps: addr u ===> )( Datenblock senden )
 3        DUP PUT OVER + SWAP
 4        ?DO   I C@ CPUT   LOOP ;
 5
 6 : GETS ( ps: addr u1 ===> u2 )( Datenblock empfangen )
 7        ( maximale Pufferlaenge u1, Laenge des Blocks u2 )
 8        GET DUP >R UMIN R@ OVER - -ROT OVER + SWAP
 9          ?DO   CGET I C!   LOOP
10        0 ?DO   CGET DROP   LOOP   R> ;
11
12
13
14
15
```

Kopplung von dBase III mit Supercalc 3

Lutz Göhrs, Stahnsdorf

Grundlage zur Einbindung von beliebigen Fremdprogrammen in dBase III ist der Befehl RUN ⟨name⟩. Mit diesem Befehl können COM, EXE und BAT-Dateien von der dBase III-Programmebene oder aus dBase III-Programmen zur Ausführung gebracht werden. Nach Ablauf des gerufenen Programms setzt dBase III seine Arbeit unmittelbar nach dem RUN-Kommando fort; entweder mit dem Prompt oder mit dem nächsten Befehl des dBase III-Programms. Dabei ist zu beachten, daß die Ausführung des RUN-Befehls Speicherplatz in Höhe von 17 KByte plus Größe des gerufenen Programms zusätzlich zu den 256 KByte von dBase III benötigt. Zur Ausführungszeit muß die Datei COMMAND.COM verfügbar sein. Von besonderem Interesse ist dabei die Einbindung von Kalkulationsprogrammen wie Supercalc 3, denn damit lassen sich mit geringem Aufwand die Vorteile der Tabellenberechnung inklusive Geschäftsgrafik mit dBase III verbinden, ohne den Aufwand der Einarbeitung und Bedienung neuer integrierter Programmsysteme in Kauf nehmen zu müssen. Die Einbindung von Kalkulationsprogrammen in dBase III entbindet durch deren automatischen Ablauf den Nutzer von speziellen Bedienkenntnissen des Kalkulationsprogramms.

Im folgenden Beispiel wird gezeigt, wie die Daten der Datenbank ARTIKEL, deren Struktur und aktueller Inhalt aus den Bildern 1 und 2 hervorgehen, in die SC3-Tabelle BESTAND (Bild 3) übertragen werden. Nach der Berechnung wird die Tabelle ausgedruckt. Die Ergebnisse in der Spalte Summe werden als Grundlage für weitere Auswertungen in die Datenbank BESTAND übernommen (Bild 4). Das Beispielprogramm in Bild 5 erzeugt eine SC3-Kommandodatei SCKOPP.XQT (Bild 6), in der alle notwendigen Anweisungen wie

- Laden der Tabelle
- Eintragung der aktuellen Werte
- Berechnen und Drucken

enthalten sind. Hinweise zum Programm sind als Kommentare gekennzeichnet. Die SC3-Tabelle sollte mit den Optionen

Bild 1 Struktur der dBase III-Datei ARTIKEL

```
Datenbankstruktur        - ARTIKEL.dbf
Anzahl der Datensätze    -        7
Letztes Änderungsdatum   - 07.07.88
Feld   Feldname     Typ          Länge    Dez        Bemerkung
   1   BEZEICHNG    Zeichen        10
   2   ARTIKEL_NR   Zeichen         6
   3   MON1         Numerisch       6                Bestand Januar
   4   MON2         Numerisch       6                       Februar
   5   MON3         Numerisch       6                       usw.
   6   MON4         Numerisch       6
   7   MON5         Numerisch       6                       bis
   8   MON6         Numerisch       6                       Juni
   9   SC_ZEILE     Zeichen         2                Zeilennummer
** Gesamt **                       55                in SC-Tabelle
```

Bild 2 Inhalt der dBase III-Datei ARTIKEL

#	BEZEICHNG	ARTIKEL_NR	MON1	MON2	MON3	MON4	MON5	MON6	SC_ZEILE
1	Deckel A	422563	268	364	470	322	287	136	7
2	Deckel B	422565	790	790	645	689	494	388	8
3	Hülle	177328	0	50	80	100	120	150	10
4	Knopf	255218	134	150	147	120	80	118	6
5	Schnalle	932850	364	467	591	288	412	725	12
6	Senkel	382616	48	60	147	92	51	17	11
7	Strumpf	833371	21	41	37	29	16	10	9

Bild 3 Struktur der dBase III-Datei BESTAND

```
Datenbankstruktur        - bestand.dbf
Anzahl der Datensätze    -       0
Letztes Änderungsdatum   - 07.07.88
Feld   Feldname     Typ          Länge    Dez
   1   BEZEICHNG    Zeichen        10
   2   ARTIKEL_NR   Zeichen         7
   3   BESTAND      Numerisch      10
** Gesamt **                       28
```

Bild 4 vorbereitete SC3-Tabelle BESTAND

```
   !    A    ::    B    :: C  :: D  ::  E  ::  F  ::  G  ::  H  ::  I  :
1   Beispieltabelle zur dBASE3-SC-Kopplung
2
3                                Bestand im Monat
4   Bezeichnung   Artikelnr Januar  Feb.   März  April   Mai   Juni   Summe
5   ------------------------------------------------------------------------
6                                                                        0
7                                                                        0
8                                                                        0
9                                                                        0
10                                                                       0
11                                                                       0
12                                                                       0
13  ------------------------------------------------------------------------
14                Summe:        0      0      0      0      0      0      0
15            Mittelwert:       0      0      0      0      0      0      0
16  ========================================================================
```

```
* Beispielprogramm zur Kopplung dBASE3 --> SuperCalc
* Annahmen: dBASE3 im Verzeichnis \DBASE\DBASE3
*       SuperCalc3 im Verzeichnis \SC3
SET TALK OFF
SET CONSOLE OFF
USE ARTIKEL
* SC3-Kommandodatei SCKOPP.XQT eröffnen
SET ALTERNATE TO SCKOPP.XQT
SET ALTERNATE ON
* Lade-Kommando für die SC3-Tabelle
? "/lBESTAND,a"
* Tabellencursor durch Sprungbefehl setzen
* Bezeichnung, Artikelnr und Monatswerte eintragen
DO WHILE .NOT. EOF()
   zeile=sc_zeile
   ? "=B&zeile"
   ? artikel_nr
   ? "=A&zeile"
   ? bezeichng
   z=1
   DO WHILE Z <= 6
      ? "=",chr(ASC("C")+z-1),zeile
      cz=str(z,1)
      ? mon&cz
      z=z+1
   ENDDO
   SKIP
ENDDO
* Tabelle berechnen, Rand ausblenden und drucken
? "!"
? "/GB"
* Output,Display,All,Printer
? "/odall,p"
* Für die Übernahme der Bestandssummen in die dBASE-Datei BESTAND
* muß die Tabellenform der Dateistruktur angepaßt werden.
* Dazu sind folgende Arbeitsgänge notwendig:
* 1. Spalten A, B und I hintereinander ordnen
*    SC3-Kommando: Move,Columne,I,C
? "/MCI,C"
* 2. Spaltenbreite der Spalten A,B und C auf dBASE-Feldlänge ändern
? "/FCA,10"
? "/FCB,7"
? "/FCC,10"
* Ausgabe der Tabelle als ASCII-Datei "BESTAND.TXT"
IF FILE("BESTAND.TXT")
   * Wenn ASCII-Datei bereits vorhanden, dann Backup
   ? "/odA6:C12,d\DBASE\DBASE3\BESTAND.TXT,b"
ELSE
   ? "/odA6:C12,d\DBASE\DBASE3\BESTAND.TXT"
ENDIF
* SC3 beenden
? "/QY"
* SC3-Kommandodatei schließen
CLOSE ALTERNATE
* BAT-Datei für Verzeichniswechsel und Aufruf schreiben
SET ALTERNATE TO SCKOPP.BAT
SET ALTERNATE ON
? "CD \SC3"
? "SC3 \DBASE\DBASE3\SCKOPP.XQT"
? "CD \DBASE\DBASE3"
CLOSE ALTERNATE
SET CONSOLE ON
* SuperCalc starten durch Aufruf der .BAT-Datei
RUN SCKOPP
* Seitenvorschub nach SC-Tabelle durch dBASE 3
EJECT
* Wenn dBASE 3 und SuperCalc im selben Verzeichnis sind
* kann das Programm etwas vereinfacht werden.
* Es braucht keine .BAT-Datei erzeugt zu werden.
* Der Aufruf lautet dann: "RUN SC3 SCKOPP.XQT"
* Übernahme der Tabellenergebnisse in die Datenbank BESTAND
USE BESTAND
ZAP
APPEND FROM BESTAND.TXT SDF
CLOSE DATABASE
RETURN
```

Bild 5 Programm zur Kopplung dBase III → SuperCalc3 → dBase III

```
Seite 1
/lBESTAND,a
=B7
422563
=A7
Deckel A
= C 7
  268
= D 7
  364
= E 7
  470
= F 7
  322
= G 7
  287
= H 7
  136
=B8
422565
=A8
Deckel B
= C 8
  790
= D 8
  790
= E 8
  645
= F 8
  689
= G 8
  494
= H 8
  388
=B10
177328
=A10
Hülle
= C 10
    0
= D 10
   50
= E 10
   80
= F 10
  100
= G 10
  120
= H 10
  150
=B6
255218
=A6
Knopf
= C 6
  134
= D 6
  150
= E 6
```

```
Seite 2
   147
= F 6
   120
= G 6
    80
= H 6
   118
=B12
932850
=A12
Schnalle
= C 12
   364
= D 12
   467
= E 12
   591
= F 12
   288
= G 12
   412
= H 12
   725
=B11
382616
=A11
Senkel
= C 11
    48
= D 11
    60
= E 11
   147
= F 11
    92
= G 11
    51
= H 11
    17
=B9
833371
=A9
Strumpf
= C 9
    21
= D 9
    41
= E 9
    37
= F 9
    29
= G 9
    16
= H 9
    10
!
/GB
/odall,p
```

```
Seite 3
/MCI,C
/FCA,10
/FCB,7
/FCC,10
/odA6:C12,d\DBASE\DBASE3\BESTAND.TXT,b
/QY
```

Bild 6 vom dBase III-Programm erzeugte SC3-Kommandodatei und .BAT-Datei

Bezeichnung	Artikelnr	Januar	Feb.	März	April	Mai	Juni	Summe
				Bestand im Monat				
Knopf	255218	134	150	147	120	80	118	749
Deckel A	422563	268	364	470	322	287	136	1847
Deckel B	422565	790	790	645	689	494	388	3796
Strumpf	833371	21	41	37	29	16	10	154
Hülle	177328	0	50	80	100	120	150	500
Senkel	382616	48	60	147	92	51	17	415
Schnalle	932850	364	467	591	288	412	725	2847
	Summe:	1625	1922	2117	1640	1460	1544	10308
	Mittelwert:	232,14	274,57	302,43	234,29	208,57	220,57	245,4286

Bild 7 durch Supercalc3 gedruckte Tabelle

```
. use bestand
. list
Satz #  BEZEICHNG   ARTIKEL_NR   BESTAND
     1  Knopf         255218         749
     2  Deckel A      422563        1847
     3  Deckel B      422565        3796
     4  Strumpf       833371         154
     5  Hülle         177328         500
     6  Senkel        382616         415
     7  Schnalle      932850        2847
```

Bild 8 Inhalt der dBase III-Datei Bestand nach Programmablauf

```
CD \SC3
SC3 \DBASE\DBASE3\SCKOPP.XQT
CD \DBASE\DBASE3
```

Bild 9 Inhalt der Datei SCKOPP.BAT

- **Nächst = OFF**
- **Manuell = ON**

gespeichert sein, um einen möglichst schnellen Programmablauf zu erreichen (Global-Befehl).
Während des Programmablaufs, der insgesamt 55 Sekunden dauerte, wurde durch SuperCalc3 die in Bild 7 gezeigte Tabelle gedruckt. Den Inhalt der Datei BESTAND zeigt Bild 8, und Bild 9 zeigt die Datei SCKOPP.BAT.

⊠ KONTAKT ☏

VEB Lufttechnische Anlagen Stahnsdorf, Ruhlsdorfer Straße 23, Stahnsdorf, 1533; Tel. 6262

Komfortable Menüs für die Tabellenberechnung

Dr. Thomas Streubel, Wolfgang Blume, Berlin

Bei Anwendung der Standardsoftware Supercalc zur Tabellenberechnung können bekanntlich zur automatisierten Abarbeitung von festen Kommandofolgen und Berechnungen, insbesondere für ungeübte Anwender XQT-Kommandodateien eingesetzt werden. Eine wünschenswerte bequeme Menüsteuerung, die den Nutzerkreis erheblich erweitern könnte, wird dabei nur sehr umständlich und mit großem Aufwand erreicht. Im folgenden wird eine sehr einfache und wirkungsvolle Methode vorgestellt, mit deren Hilfe schnell eine höheren Programmiersprachen adäquate Menügestaltung erreicht werden kann.
Die Selektion von Programmen im Menü basiert auf der Tatsache, daß zum Aufruf einer XQT-Kommandodatei der Name der gewünschten Datei entsprechend den CP/M-Konventionen mit Dateityp erfolgen muß. Bei der Syntaxanalyse werden von der Typbezeichnung **XQT** nur die ersten beiden Zeichen erkannt, während jedoch für den danach erfolgenden Aufruf alle drei benötigt werden. Das bietet die Möglichkeit, in einer Kommandodatei DEMO.XQT (Bild 1) im System durch einen unvollständigen Programmaufruf wegen eines fehlenden **T** in der Dateityp-Bezeichnung einen (Fehler-) Zwangshalt zu provozieren. Der zuvor im Programm realisierte Aufruf eines erklärenden Textes der Datei DEMO.CAL (Bild 2) illustriert gut die sich anschließende Auswahlmöglichkeit durch den Nutzer, der nach diesem Halt das fehlende dritte Zeichen entsprechend dem Menütext hinzufügen und diese Eingabe mit der **ET**-Taste abschließen kann. Das Tabellenberechnungsprogramm gelangt somit bei der anschließenden syntaktischen Prüfung zum Ergebnis, daß der damit spezifizierte (Unter-) Programmaufruf (entweder DATEI.XQA, DATEI. XQB ... DATEI.XQL) korrekt ist und damit nun auch wirklich ausführbar wird.
Nach diesen Manipulationen wird letztendlich das gewünschte Programm geladen. Es kann sowohl ein Unterprogramm als auch ein ähnlich strukturiertes Untermenü sein. Für diesen Fall sichert eine Programmschleife die Rückkehr ins aufrufende übergeordnete Menü durch einen entsprechenden Programmaufruf nach dem selben Muster. Im Falle einer möglichen Fehleingabe erscheint die Fehlermeldung „Datei nicht ladbar", und das System verbleibt in der Abfrage. Auf diese Art und Weise kann eine streng hierarchisch aufgebaute und geschlossene Menüführung entstehen. Sie arbeitet auch bei Fehleingaben absturzsicher, was besonders für die Nutzung durch den ungeübten Nutzer sehr wichtig ist.
Mit Erfolg wird dieses Prinzip bereits in Software-Schulungen vorgestellt und in mehreren konkreten Anwendungen genutzt.

```
/ZY              Arb.-Bereich löschen
/GB              Begrenzung aus
/GM              manu. Berechn.-Start
/LB:DEMO,A       Text laden u. zeigen
/GB              evtl. Begrenzung aus
/XB:DATEI.XQ     Halt, was nun ??!!!!
```

Bild 1 Beispielprogrammvarianten, die einen Zwangshalt erzeugen

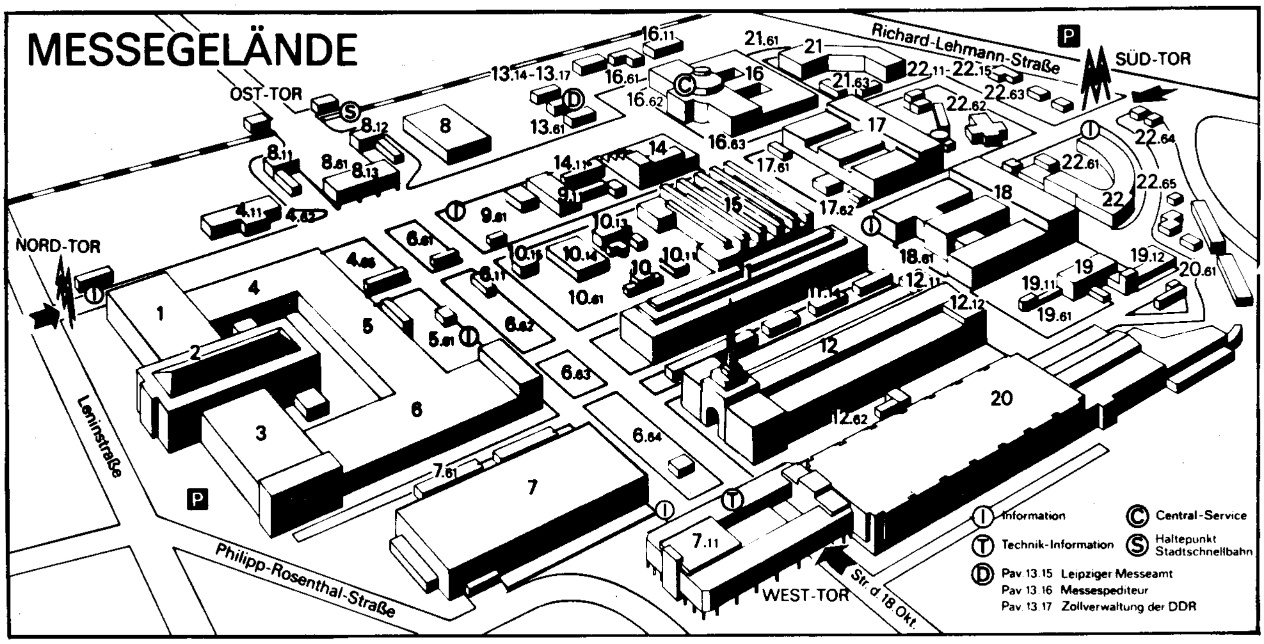
```
            ======= Menü der Monatsabrechnungen =======

   "A"-Taste   --->  Abrechnung JANUAR      (DATEI.XQA)
   "B"-Taste   --->  Abrechnung FEBRUAR     (DATEI.XQB)
   "C"-Taste   --->  Abrechnung MAERZ       (DATEI.XQC)
   "D"-Taste   --->  Abrechnung APRIL       (DATEI.XQD)
   "E"-Taste   --->  Abrechnung MAI         (DATEI.XQE)
   "F"-Taste   --->  Abrechnung JUNI        (DATEI.XQF)
   "G"-Taste   --->  Abrechnung JULI        (DATEI.XQG)
   "H"-Taste   --->  Abrechnung AUGUST      (DATEI.XQH)
   "I"-Taste   --->  Abrechnung SEPTEMBER   (DATEI.XQI)
   "J"-Taste   --->  Abrechnung OKTOBER     (DATEI.XQJ)
   "K"-Taste   --->  Abrechnung NOVEMBER    (DATEI.XQK)
   "L"-Taste   --->  Abrechnung DEZEMBER    (DATEI.XQL)
                                    ---------
 "0"-Taste   --->  Rueckkehr ins Hauptmenue   (DATEI.XQ0)

                 ****** Bitte waehlen ******

< A1
==> Dateiname oder <return> fuer Verzeichnis
 19>/X(Execute),B:DATEI.XQ_
```

Bild 2 Bildschirminhalt während des Zwangshalts



Messeführer LFM '90

Mit dem Messewegweiser auf dieser Seite wollen wir alle an der Computertechnik Interessierten beim Besuch der Technischen Messe unterstützen. Wir haben daher für Sie die Aussteller – soweit sie uns bis zu unserem Redaktionsschluß bekannt waren – aus den Messekatalog-Rubriken *Informations- und Kommunikationstechnik* sowie *Automatisierungstechnik – CAD/CAM* übersichtlich aufgelistet – zum einen nach Hallen geordnet, zum anderen in alphabetischer Reihenfolge. Die Palette reicht daher von Bauelementeproduzenten über Computer- und Peripheriehersteller bis zu Softwareanbietern. Eine Gewähr für die Richtigkeit der Angaben können wir Ihnen leider nicht geben, hoffen aber, daß Ihnen unser kleiner Service dennoch ein nützliches Hilfsmittel sein wird.

	Stand
Halle 2	
Mannesmann Kienzle GmbH (BRD)	112
Halle 3	
Fexima Oy (Finnland)	205
Halle 5	
Agfa Gevaert AG (BRD)	310
Hoechst AG (BRD)	340
Halle 6	
VEB Organisation und Rechentechnik Karl-Marx-Stadt (im VE Kombinat Mechanisierung Karl-Marx-Stadt)	
Halle 7	
Akademie der Wissenschaften der DDR	215
Hoch- und Fachschulwesen der DDR	217
Halle 14	
Kombinat VEB Carl Zeiss JENA	
Halle 15	
VEB Kombinat Automatisierungsanlagenbau Berlin	
Bull AG (BRD)	130
Kombinat VEB Elektro-Apparate-Werke Berlin	
Honeywell Regelsysteme GmbH (BRD)	130
IBM ROECE INC. Vienna-Branch. (Österreich)	108
Kombinat VEB Keramische Werke Hermsdorf	
VEB Kombinat Mikroelektronik Erfurt	
VEB Kombinat Nachrichtenelektronik Berlin	
Philips International B.V. (Niederlande)	27
VEB Kombinat Robotron Dresden	
Schlumberger Technologies GmbH (Österreich)	94
Siemens AG (BRD)	44
Standard Elektrik Lorenz AG (BRD)	104/111
Tektronix GmbH (BRD)	47
VIDEOTON Industrie-Außenhandels-AG (Ungarn)	33
Wandel & Goltermann GmbH (BRD)	57
Wapo-electronic GmbH (Berlin (W))	128
VEB Zentrum für Forschung und Technik Dresden (im VEB Kombinat Spezialtechnik Dresden)	
Halle 16	
CUBAELECTRONICA (Kuba)	
Halle 17	**Stand**
A.B. Dick Company (BRD)	40
Aristo Graphic Systeme GmbH (Berlin (W))	54
ATARI Computer GmbH (BRD)	58
Commodore Büromaschinen GmbH (BRD)	96
EPSON Deutschland GmbH (BRD)	66
Ensch S.a.r.l., Ensch Computer Luxembourg	37
Inverdata Electronics GmbH (BRD)	41
METRONEX G.m.b.H. (Polen)	90
MINOLTA AUSTRIA-Export Ges. m.b.H. (Österreich)	60
Ungarische Optische Werke MOM	94
Rank Xerox Ltd. (Großbritannien)	18
3M (East) AG (Schweiz)	61
Halle 18	
ABC Computer Technik Röstges (BRD)	265
Electronum (Rumänien)	132
Graphtec GmbH (BRD)	122
IPD Electronic GmbH (BRD)	208
Mütec, Meß- und Regeltechnik GmbH (BRD)	104
Schiele Industriewerke GmbH (BRD)	215
Spectronex AG (Schweiz)	212
Toshiba Corporation (Japan)	8
Transcommerz Industriewaren Handels-GmbH (Berlin (W))	10
Westinghouse Electric GmbH Controlmatic Division (Österreich)	200
Halle 20	
VEB Werkzeugmaschinenkombinat Karl-Marx-Stadt	

	Halle	Stand
ABC Computer Technik Pöstges (BRD)	18	265
A.B. Dick Company (BRD)	17	40
Agfa Gevaert AG (BRD)	5	310
Akademie der Wissenschaften der DDR	7	215
Aristo Graphic Systeme GmbH (Berlin (W))	17	54
ATARI Computer GmbH (BRD)	17	58
VEB Kombinat Automatisierungs-anlagenbau Berlin	15	
Bull AG (BRD)	15	130
Commodore Büromaschinen GmbH (BRD)	17	96
CUBAELECTRONICA (Kuba)	16	
Electronum (Rumänien)	18	132
Kombinat VEB Elektro-Apparate-Werke Berlin	15	
Ensch S.a.r.l, Ensch Computer Luxembourg	17	37
EPSON Deutschland GmbH (BRD)	17	66
Fexima Oy (Finnland)	3	205
Graphtec GmbH (BRD)	18	122
Hoch- und Fachschulwesen der DDR	7	217
Hoechst AG (BRD)	5	340
Honeywell Regelsysteme GmbH (BRD)	15	130
IBM ROECE INC. Vienna-Branch. (Österreich)	15	108
Inverdata Electronics GmbH (BRD)	17	41
IPD Electronic GmbH (BRD)	18	208
Kombinat VEB Keramische Werke Hermsdorf	15	
Mannesmann Kienzle GmbH (BRD)	2	112
METRONEX G.m.b.H. (Polen)	17	90
VEB Kombinat Mikroelektronik Erfurt	15	
MINOLTA AUSTRIA-Export Ges. m.b.H. (Österreich)	17	60
Ungarische Optische Werke MOM	17	94
Mütec, Meß- und Regeltechnik GmbH (BRD)	18	104
VEB Kombinat Nachrichtenelektronik Berlin	15	
VEB Organisation und Rechentechnik Karl-Marx-Stadt (im VE Kombinat Mechanisierung Karl-Marx-Stadt	6	
Philips International B.V. (Niederlande)	15	27
Rank Xerox Ltd. (Großbritannien)	17	18
VEB Kombinat Robotron Dresden	15	
Schiele Industriewerke GmbH (BRD)	18	215
Schlumberger Technologies GmbH (Österreich)	15	94
Siemens AG (BRD)	15	44
Spectronex AG (Schweiz)	18	212
Standard Elektronik Lorenz AG (BRD)	15	104/111
Tektronix GmbH (BRD)	15	47
Toshiba Corporation (Japan)	18	8
Transcommerz Industriewaren Handels-GmbH (Berlin (W))	18	10
VIDEOTON Industrie-Außenhandels-AG (Ungarn)	15	33
Wandel & Goltermann GmbH (BRD)	15	57
Wapo-electronic GmbH (Berlin (W))	15	128
VEB Werkzeugmaschinenkombinat Karl-Marx-Stadt	20	
Westinghouse Electric GmbH Controlmatic Division (Österreich)	18	200
Kombinat VEB Carl Zeiss JENA	14	
VEB Zentrum für Forschung und Technik (im VEB Kombinat Spezialtechnik Dresden)	15	
3M (East) AG (Schweiz)	17	61

Unser Tip: Lesertreffpunkt

Gemeinsam mit unserer Partnerredaktion c't wollen wir uns an zwei Tagen der Messe Ihren Fragen, Wünschen und Kritiken stellen. Dazu beabsichtigen wir, am 12. und 13. März für jeweils mehrere Stunden die Technik-Information der KDT in Halle 7.11 zu nutzen. Gesichert ist das Vorhaben zwar noch nicht, aber wenn es ihnen keine großen Umstände bereitet, schauen Sie doch einfach mal vorbei. Wir würden uns freuen.

Ihre Redaktion MP

Computer-Club

Wordstar nutzt alle Drucksteuerzeichen

Die vielfältigen Schriftgestaltungsmöglichkeiten moderner Nadel-Matrix-Drucker (besonders des FX 1000) werden durch Wordstar, Version 2.x, nicht direkt ausgeschöpft. Von Hause aus werden durch Wordstar nur vier Steuerzeichen (^E, ^R, ^Q, ^W) für den Nutzer zur Verfügung gestellt. Auch bei Nutzung weiterer (^A, ^N, ^T, ^V, ^Y) ergeben sich nicht wesentlich mehr Möglichkeiten, da einige Zeichen nicht unabhängig voneinander genutzt werden können. Greift man die Idee aus /1/ auf und belegt eines dieser Zeichen – sinnfälligerweise ^E – lediglich mit dem vom Drucker als ESCape und damit Beginn einer Steuersequenz verstandenen Code 1BH, sind damit alle die zugänglich, die aus druckbaren ASCII-Zeichen bestehen. Als Parameter sind darüber hinaus noch 0 und 1 zugelassen, da der Drucker in diesem Fall sowohl die ASCII-Codes 00H und 01H als auch die ASCII-Zeichen 0 (30H) und 1 (31H) akzeptiert. Im Bild 1 sind einige der Möglichkeiten mit einem so installierten Wordstar zusammengestellt.

Nach der Eingabe von ^PE werden die restlichen Zeichen wie normaler Text eingegeben. Dies bedeutet allerdings, daß Wordstar sie auch so behandelt, obwohl sie beim Druck nicht erscheinen, denn der Drucker interpretiert sie korrekt. Daher ist bei eingeschaltetem Blocksatz die Ausrichtung mit der Hand vorzunehmen und entsprechend der Anzahl der Zeichen über den rechten Rand hinaus zu schreiben. Auch sollte man häufig benutzte Sequenzen auf die *normalen* Druckersteuerzeichen von Wordstar legen, damit dieser Nachteil dadurch gemildert wird, daß die zusätzlichen Sequenzen überwiegend global oder nur auf eine Zeile als ganzes (z.B. Überschriften) angewandt werden.

Die Installierung kann entweder mit einem entsprechenden Programm, einem Debugger oder Dienstprogrammen wie POWER, TOOL, DIDOC erfolgen. Für den letztgenannten Fall sind in der nachfolgenden Tabelle die entsprechenden Patch-Adressen zusammengestellt.

Die Länge der Sequenz für die Steuerzeichen beträgt 4 Byte, die von Anfangs- und Endsequenz 15 Byte.

Steuerzeichen	WS.COM /1/ 8 Bit	WS.CMD 16 Bit
^A	06B5	076B
^N	06BA	0770
^T	06BF	0775
^V	06C4	077A
^Q	06C9	077F
^W	06CE	0784
^E	06D3	0789
^R	06D8	078E
^Y (ein)	06DD	0793
^Y (aus)	06E2	0798
Anfangssequenz (Druckerinitial.)	06E7	079D
Endsequenz	06F8	07AE

Hinzu kommt in allen Fällen ein Führungsbyte, das die tatsächliche Länge der jeweiligen Sequenz enthält (01H bis 04H bzw. 01 bis 0FH). Im vorliegenden Fall würde für ^E ab Adresse 0789 beim 16-Bit-WS folgende Sequenz stehen: 011B 00 00 00.

Bei höheren Versionen von Wordstar unter MS-DOS benutzt man am besten das Installationsprogramm WSCHANGE.

Außer den dargestellten Einzelfunktionen existieren noch viele Kombinationsmöglichkeiten.

Literatur

/1/ Floriani, D.: Wordstar maßgeschneidert. mc, München (1985) 7

Eberhard Bauer

```
Bildschirm-                    Druckbild
darstellung
-----------------------------------------------------------------------

  ^A        alternativer Zeichenabstand (17 cpi bei K 6314, 20 cpi bei FX-1000)
  ^N        Standardzeichenabstand (hier 12 cpi)
  ^Q        Hoch-/Tiefstellen aus
  ^R        Tiefstellen
  ^S        Unterstreichen ein/aus (besser Y benutzen!)
  ^T        alternativer (US-ASCII) Zeichensatz ({ ¦ } [ \ ] ~ @)
  ^V        deutscher Zeichensatz (ä ö ü Ä Ö Ü ß §)
  ^W        Hochstellen
  ^Y        Unterstreichen ein/aus
  ^EE       Fettdruck (K6314: 10 cpi, FX1000/LX-86: gewählter Abstand bleibt erhalten)
  ^EE^EG    Fett- und Doppeldruck kombiniert
  ^EM       Elite-Schrift(12cpi)
  ^EP       Pica-Schrift (10 cpi)
  ^EW1      Breitdruck
  ^E0       Zeilenabstand 1/8", d. h.
            einzeilig
            einzeilig
  ^E1       Zeilenabstand 7/72", d. h.
            engzeilig
            engzeilig
  ^E2       Zeilenabstand 1/6" (Standard)
  ^E8       Schaltet den Papierendesensor aus.

Zusätzlich bieten der FX-1000 und der LX-86 folgende Funktionen:

  ^Ex1      NLQ-Schrift
  ^E4       Kursivdruck
  ^E%1      Benutzerdefinierte Zeichen aktivieren (die vor dem Start von WS
            geladen werden müssen). So wird beispielsweise "a a a" zu
            "♥ ♥ ♥".
  ^E%0      zurückschalten auf normalen Zeichensatz

Der FX-1000 kann darüber hinaus:

  ^Ek0      NLQ-Schriftart ROMAN
  ^Ek1      NLQ-Schriftart SANS SERIF
  ^Ep1      Proportionaldruck
  ^Ep0      Proportionaldruck aus
  ^Ew1      Doppelte Zeichenhöhe
  ^Ew1^EW1  doppelte Zeichenhöhe und -breite

  ^Et1^E>   aktiviert den Grafikteil des Zeichensatzes 3 und 4 (Codes > 128)
            und setzt intern das MSB auf 1, so daß anstelle der eingegebenen
            Zeichen die gedruckt werden, deren Code um 128 größer ist. Bei-
            spielsweise wird 'f' zu 'µ'. Mit dieser Funktion sind leider nur
            die Grafikzeichen ab Codenummer 160 zugänglich, da die zwischen
            128 und 159 keine druckbaren Entsprechungen im Standard-ASCII-
            Teil des Zeichensatzes haben (Code 0 - 31).
  ^E#       hebt die Vorgabe des MSB wieder auf.
```

***Bild 1** Drucksteuerung durch Wordstar nach dem Patchen*

• Alle Angaben beziehen sich auf den erweiterten EPSON-ESC/P-Standard

• Da für die Nutzung der Hoch- und Tiefstellfunktion insgesamt 3 Sequenzen erforderlich sind, wurden auch drei Steuerzeichen damit belegt.

• Es empfiehlt sich, Standardeinstellungen in die Anfangssequenz mit aufzunehmen. Dies betrifft beim FX-1000 auch die Sequenz ^Et1 (exakter müßte es heißen: ESC t 1 oder 1BH 74H 31H), so daß nur noch das MSB (most significant bit, deutsch: höchstwertiges Bit) vorgegeben werden muß (^E>), wenn Zeichen aus dem Grafikteil gedruckt werden sollen.

Druckausgabe mit Turbo-Pascal umleiten

Der DCP-Befehl MODE bietet die Möglichkeit, die Druckerausgabe von der parallelen (Standard-) Druckerschnittstelle auf die serielle Schnittstelle umzuleiten. Mit dem folgenden Verfahren ist es möglich, solche Programme mit Turbo-Pascal 4.0 zu erstellen. Auf diese Weise können leicht verschiedene Protokollarten und Druckeranpassungen programmiert werden.

Der externe Befehl MODE bietet ebenfalls die Möglichkeit, die parallele und die serielle Schnittstelle zu konfigurieren. Die Umlenkung von der parallelen auf die serielle Schnittstelle erfolgt, indem der Softwareinterrupt **Int17** auf den residenten Teil von MODE geleitet wird. Dort werden die Übergabeparameter verändert und die Daten über **Int14** an die serielle Schnittstelle gesandt. Unter DCP gab es jedoch auf dem A 7150 mit der IFSS-Schnittstelle (Com2) gewisse Probleme:

- Es wird kein Softwareprotokoll unterstützt und damit keine Anpassung an die Druckgeschwindigkeit ermöglicht. Die Folge sind nichtgedruckte Zeichen.
- Die Funktion **Initialisieren des INT17** wird nicht korrekt an den INT14 übergeben und führt zur Fehlermeldung des Betriebssystems.

Sollen höhere Programmiersprachen für die vorgestellte Art der systemnahen Programmierung genutzt werden, müssen diese folgende Möglichkeiten bieten:

- Zugriff auf Softwareinterrupts
- Programmieren speicherresidenter Interruptroutinen.

Turbo-Pascal 4.0 bietet dies mit den Prozeduren INTR, KEEP und der Deklaration von Prozeduren als Interruptroutinen.

Bild 1 zeigt einen Treiber für den Drucker K 6311 unter Verwendung des DC1/DC3-Protokolls. Das Hauptprogramm führt die Initialisierung der IFSS-Schnittstelle durch, wobei folgende Werte eingestellt werden:

- Übertragungsrate 9600 Baud
- ungerade Parität
- 1 Stopbit
- 7 Bit pro Zeichen.

Die Kodierung dieser Werte ist /1/ zu entnehmen.

Anschließend wird die Interrupttabelle verändert. Der Softwareinterrupt INT17 zeigt nun auf die Prozedur **INT17neu**; das Programm wird speicherresident beendet.

Nutzt nun ein Programm den INT17, so führt die Prozedur INT17neu die Anpassung der Übergabe- und Rückgabeparameter nach Tafel 1 durch. Anschließend wird das Zeichen mit **INT14** an die serielle Schnittstelle gesandt. Die Senderoutine kontrolliert den Leitungsstatus und verhindert damit ein Überschreiben von Daten. Werden DC3/DC4-Signale empfangen, warten die entsprechenden Routinen auf DC1. Nach einer Fehler-

Tafel 1 Gegenüberstellung der Parameter von INT17 und INT14 (speziell für IFSS)

Funktion		INT17 Registerinhalt	INT14 Registerinhalt
Zeichen senden	Aufruf	AH 00 AL Zeichen DX 00	AH 01 AL Zeichen DX Kanalnummer 01
	Rückkehr	AH Status	AH Leitungsstatus
Initialisieren	Aufruf	AH 01 DX 00	AH 00 AL Parameter DX Kanalnummer 01
	Rückkehr	AH Status	AH Leitungsstatus
Statusabfrage	Aufruf	AH 02 DX 00	AH 03 DX Kanalnummer 01
	Rückkehr	AH Status	AH Leitungsstatus
Zeichen empfangen	Aufruf		AH 02 DX Kanalnummer 01
	Rückkehr		AH Leitungsstatus AL Zeichen
Bedeutung des Status- bzw. Leitungsstatus-Bytes		09 Gerätefehler 90 Gerät empfangsbereit	80 Time out 01 Empfangsdaten vorhanden 40 Sendepuffer leer

```
  0:
  1:
  2: PROGRAM MODEplus;
  3: uses dos;
  4: {$M 1024,0,0 }
  5:
  6: VAR
  7:      i               : integer;
  8:      Empfbyte        : byte;
  9:      Empfang,Senden  : boolean;
 10:      Regs            : registers;
 11:
 12: CONST
 13:    Geraetestatus : ARRAY [1..4] OF byte =($1b,$5b,$35,$6e);
 14:    Totschleife = 10000;
 15:    DC1 = $11;
 16:    DC3 = $13;
 17:    DC4 = $14;
 18:
 19: PROCEDURE INT17neu(ax,bx,cx,dx,si,di,ds,es,bp : word);
 20:  interrupt;
 21:
 22:  PROCEDURE status;
 23:    BEGIN
 24:      Empfang:=false;
 25:      Senden:=false;
 26:      regs.ah:=03;
 27:      regs.dx:=01;
 28:      intr($14,regs);
 29:      IF (regs.ah AND 01) = 1 THEN Empfang:=true;
 30:      IF (regs.ah AND $40) = $40 THEN Senden:=true;
 31:    END;
 32:
 33:  PROCEDURE ByteLesen;
 34:    BEGIN
 35:      regs.dx:=01;
 36:      regs.ah:=02;
 37:      intr($14,regs);
 38:      Empfbyte:=regs.al AND $7f;
 39:    END;
 40:
 41:  PROCEDURE WartenaufDC1;
 42:    BEGIN
 43:      i:=0;
 44:      REPEAT
 45:        i:=i+1;
 46:        status;
 47:        IF Empfang THEN ByteLesen;
 48:      UNTIL ((Empfbyte=DC1) OR (i=Totschleife));
 49:    END;

 51:
 52:   BEGIN   { INT17neu }
 53:    regs.ax:=AX;
 54:    regs.dx:=01;
 55:    CASE regs.ah OF
 56:     00 : BEGIN            { Senden }
 57:           Status;
 58:           IF Empfang THEN ByteLesen;
 59:
 60:           IF Empfbyte=DC4 THEN
 61:              BEGIN
 62:                 FOR i:=1 TO 4 DO
 63:                   BEGIN
 64:                     regs.al:=geraetestatus[i];
 65:                     regs.ah:=01;
 66:                     intr($14,regs);
 67:                   END;
 68:                  WartenaufDC1;
 69:              END;
 70:
 71:           IF Empfbyte=DC3 THEN  WartenaufDC1;
 72:
 73:           IF Empfbyte=DC1 THEN
 74:              BEGIN
 75:                 i:=0;
 76:                 REPEAT
 77:                   i:=i+1;
 78:                   status;
 79:                 UNTIL (Senden OR (i= Totschleife));
 80:                 regs.ax:=AX;
 81:                 regs.ah:=01;
 82:                 intr($14,regs);
 83:              END;
 84:
 85:           IF (regs.ah=$80) OR (i=Totschleife) OR (Empfbyte<>DC1)
 86:              THEN  regs.ah:=$09
 87:              ELSE  regs.ah:=$90;
 88:
 89:         END;
 90:
 91:     01,02  : BEGIN      { Initialisieren,Statusabfrage }
 92:              status;
 93:              IF (regs.ah=$80)
 94:                 THEN  regs.ah:=$09
 95:                 ELSE  regs.ah:=$90;
 96:              END;
 97:
 98:    END;{case}
 99:    ax:=regs.ax;     { Parameteruebergabe}
100:   END; { INT17neu }
101:
102: BEGIN  { Main }
103:   regs.ah:=$00;        { Initialisierung IFSS }
104:   regs.al:=$ea;
105:   regs.dx:=01;
106:   intr($14,regs);
107:
108:   setintvec($17,@int17Neu);
109:
110:   writeln('MODE-plus installiert');
111:   keep(0);
112: END.
113:
114:
```

Bild 1 Programm MODEplus in Turbo-Pascal 4.0

meldung der Schnittstelle oder nach Ablauf einer bestimmten Anzahl von Schleifendurchläufen gibt das Programm eine entsprechende Fehlermeldung an das Betriebssystem zurück.
Mit diesem Verfahren ist es möglich, beliebige Übertragungsprotokolle zu ermöglichen. Durch den Einbau von Filtern können Zeichensätze verschiedener Drucker angepaßt werden. Ein Anpassen der Grafikbefehle wäre ebenfalls denkbar. Damit bietet Turbo-Pascal eine Alternative zur Assemblerprogrammierung.

Hartmut Schröter

Literatur

/1/ VEB Robotron Projekt: Anleitung für den Systemprogrammierer – DCP 1700, Ausgabe 3.20, Dresden 1988

/2/ Smode, D.: Das Große MS-DOS Profi-Arbeitsbuch. Franzis-Verlag, München 1987

/3/ Borland Inc.: Handbuch Turbo-Pascal Version 4.0 Heimsoeth Software, München 1987

Diskettenboxen

Unsere Veröffentlichung zu den Diskettenboxen in MP 11/89 hat eine derart große Nachfrage hervorgerufen, daß die Handwerkliche Buchbinderei in Bedrängnis geraten ist. Wie uns mitgeteilt wurde, wird die Situation durch Arbeitskräfteverlust noch verschärft. Sofern nicht der Berufsverband der Buchbinder Unterstützung gewährt, ist die Bearbeitung der eingegangenen Bestellungen nicht in voller Höhe gesichert. Es wird daher um Verständnis dafür gebeten, daß keine weiteren Bestellungen angenommen werden können.

MP

Bit-Funktionen unter Turbo-Pascal

Wer sich intensiver mit Turbo-Pascal befaßt, wird früher oder später auch maschinennah programmieren. In diesem Fall werden oft Funktionen zum Testen, Setzen oder Löschen von einzelnen Bits benötigt, die von Turbo-Pascal aber nicht bereitgestellt werden.
Im folgenden sollen daher vier derartige Funktionen vorgestellt werden. Da lediglich der standardmäßige Sprachumfang genutzt wird, ist der Einsatz unter allen Betriebssystemen möglich. Die Bits werden von 0 bis 15 (bzw. von 0 bis 7 bei Byte-Werten) gezählt.
Durch den Anwender muß sichergestellt werden, daß bei Manipulation von Byte-Werten nur auf existierende Bits (Bitnummer ≦7 zugegriffen wird, da sonst andere Variablen zerstört werden können oder falsche Ergebnisse entstehen.

```
PROGRAM bitfunkt;
VAR i,n:INTEGER;
    b:BYTE;

FUNCTION bit_z(i,n:INTEGER):INTEGER;
BEGIN
  bit_z:=ORD( ODD(i SHR n))
END;

FUNCTION bit_l(i,n:INTEGER):BOOLEAN;
BEGIN
  bit_l:=ODD(i SHR n)
END;

PROCEDURE bit_set(VAR i;n:INTEGER);
VAR ii:INTEGER ABSOLUTE i;
BEGIN
  ii:=ii OR (1 SHL n)
END;


PROCEDURE bit_reset(VAR i;n:INTEGER);
VAR ii:INTEGER ABSOLUTE i;
BEGIN
  ii:=ii AND NOT(1 SHL n)
END;

BEGIN
  clrscr;
  i:=-maxint;
  FOR n:=15 DOWNTO 0 DO write( bit_z(i,n));
  writeln;
  bit_reset(i,15);
  writeln( bit_l(i,15));
  bit_set(i,1);
  FOR n:=15 DOWNTO 0 DO write( bit_z(i,n));
  writeln;
  writeln(i);
  b:=127;
  bit_set(b,7);
  writeln(b);
END.
```

Funktionen BIT_Z und BIT_L

Diese Funktionen geben entweder einen Zahlenwert (0 oder 1 in BIT_Z) oder einen logischen Wert (False oder True in BIT_L) zurück.
Das Funktionsprinzip ist bei beiden dasselbe. Das gewünschte Bit gelangt durch Rechtsverschiebung in die niederwertigste Position. Bei Interpretation des Bitmusters als Integerzahl bestimmt es also, ob diese Zahl gerade oder ungerade ist. Dies kann durch die Standardfunktion ODD getestet werden. Wird als Ergebnis ein Zahlenwert verlangt, so wird das Ergebnis des Tests (FALSE oder TRUE) mit der Standardfunktion ORD in einen Zahlenwert gewandelt. Dies ist möglich, da man den Typ BOOLEAN als durch

TYPE BOOLEAN = (FALSE,TRUE)

vordefiniert betrachten kann.

Die Prozeduren BIT_SET und BIT_RESET

Durch Linksverschiebung eines einzelnen gesetzten Bits wird eine Maske erzeugt und mit der zu manipulierenden Zahl mit OR (BIT_SET) verknüpft bzw. erst bitweise negiert und dann mit AND (BIT_RESET) verknüpft.
Da die beiden Prozeduren sowohl Integerzahlen als auch Bytes verarbeiten sollen, ist die Übergabe als typfreie Variable und die anschließende Interpretation der übergebenen Variablen als Integerzahl erforderlich.

Bernd Matzke

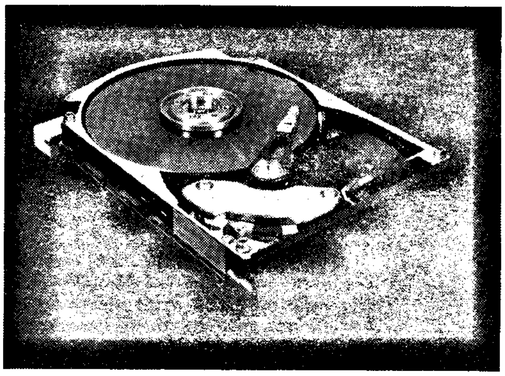
Electronic Mail integriert unterschiedliche Computerwelten

Nach Angaben von Hewlett-Packard ist die Firma der erste große EDV-Hersteller, der die Integration von Computern unterschiedlicher Betriebssysteme einschließlich Apple Macintosh, HP MPE, MS-DOS und Unix über ein Electronic Mail-System nach dem X.400-Standard anbietet.
HP OpenMail läuft unter dem Betriebssystem HP-UX, das der AT&T Unix System V Interface Definition Issue 2 entspricht. Es wurde unter Berücksichtigung der X/Open-Standards entwickelt und ist damit auf andere Unix-Versionen portierbar. HP OpenMail kommuniziert mit allen Systemen, die den X.400-Standard unterstützen.
Über HP AdvanceMail III können PC-Anwender mit HP OpenMail auf HP 9000- und HP DeskManager auf HP 3000-Systemen kommunizieren. Die neue Version hat die gleiche Benutzerschnittstelle für beide Systemumgebungen, so daß kein Umdenken erforderlich ist. Terminal-Emulation und File-Transfer zwischen der HP-Welt und Macintosh bzw. Windows bieten zwei neue Versionen der Kommunikationssoftware HP AdvanceLink. Sie ermöglichen den Macintosh- bzw. Windows-Anwendern den Zugriff auf die Electronic Mail-Dienste von HP 3000- und HP 9000-Systemen. *MP*

Neue PRISM-CPU für die Apollo-Serie 10 000

Ein Programm zur Leistungssteigerung der Personal-Supercomputer Apollo-Serie 10 000 kündigte Hewlett-Packard an. Dazu gehört auch die Entwicklung einer neuen, vollständig kompatiblen RISC-Zentraleinheit, die den Systemdurchsatz verdoppeln wird. Mitte nächsten Jahres werden alle wesentlichen Systemkomponenten der Multiprocessing-Serie 10 000 überarbeitet und deutlich verbessert: Parallelisierende und vektorisierende Compiler werden die Systemleistung verdoppeln bis vervierfachen; das Programmieren wird HP durch eine spezielle Software-Entwicklungsumgebung für Parallelverarbeitung sowie durch eine interaktive und Visualisierungs-Umgebung für technisch-wissenschaftliche Probleme unterstützen; die Hauptspeicherkapazität wird vervierfacht, die Massenspeicherkapazität versechsfacht und die Bandbreite in lokalen Netzwerken verzehnfacht.
Die neue Zentraleinheit mit PRISM-Architektur steigert die verfügbare Rechenleistung von 22 auf 44 MIPS bzw. 12 MFLOPS je Prozessor. Zusätzlich zu einer Verdoppelung der Integer-(Dhrystone-)Leistung wird die neue Zentraleinheit auch eine verbesserte Gleitkommaverarbeitung – mit dem bis zu 60fachen einer VAX 11/780 – bieten. Zu ihrer Realisierung werden in großem Umfang, beispielsweise für die Bus-Schnittstelle und den Gleitkomma-Registersatz, anwendungsspezifische Integrierte Schaltungen (ASICs) in 1-μm-Technologie eingesetzt. *MP*

Neue Harddisks von Seagate



3,5-Zoll-Laufwerk ST 3096A

Laufwerk ST 157CA für PC ATs

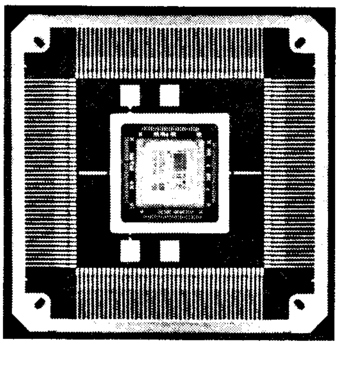
Der Festplattenhersteller Seagate Technology nutzte die SYSTEMS zur „bislang umfangreichsten Serie von Neuvorstellungen". Herausragend aus diesem Spektrum sind zum einen neue Hochleistungslaufwerke für PC ATs und 386er in Arbeitsplatzrechner- und Fileserver-Anwendungen wie die Familie ST 1144 mit bis zu 120 MByte Kapazität. Zum anderen gibt es eine als neue Harddisk-Generation bezeichnete Familie mit sechs Mitgliedern, die mit ihrer Bauhöhe von nur 1 Zoll insbesondere den Erfordernissen von Laptops entspricht. Das im Bild gezeigte Modell ST 3096A beispielsweise speichert 84 MByte (formatiert) bei 20 ms Zugriffszeit. Interessant sind auch die Steckkartenlaufwerke zum Aufrüsten von PC XTs und ATs. Die fertig konfigurierten und mit Controller versehenen Laufwerke mit bis zu 49 MByte – beim gezeigten Modell ST 157CA – werden in einen freien Slot gesteckt, ohne vorhandene Laufwerke ausbauen zu müssen.

Neuartiges Speichersystem

Von der Firma Canon wurde ein neues optisches Speichersystem für Computer vorgestellt. Die Daten befinden sich auf einem scheckkartengroßen flachen Datenträger, der in der Schreib-Lese-Einheit mittels Laser beschrieben und gelesen werden kann. Die Speicherkapazität einer Karte beträgt 2 MByte. Die Aufzeichnungsgeschwindigkeit wird mit 15,3 kByte/s, die Lesegeschwindigkeit mit 100 kByte/s angegeben. Die mittlere Zugriffszeit soll zwischen 23 ms und 2,5 s liegen.
Mit der Markteinführung wird 1991 gerechnet, während erste Muster bereits im nächsten Jahr erhältlich sein sollen. *MP*

RISC-Mikroprozessoren jetzt auch von Siemens

Die Zusammenarbeit zwischen Siemens und MIPS Computer Systems Inc. soll für 32-Bit-Mikroprozessoren nach dem RISC-Prinzip (Reduced Instruction Set Computer) neue Standards setzen. Das ist der Wunsch von Hans-Jörg Penzel (Siemens) und John Mashey (MIPS) anläßlich der Markteinführung der ersten gemeinsamen Produkte. Die Musterschaltkreise der Familie SAB-R2000A bzw. R3000 kommen aus Münchener Fertigung – nur acht Monate nach der Unterzeichnung des Kooperationsabkommens (wir berichteten darüber in MP 4/89, Seite 124). Mit so kurzer Markteinführungszeit sieht sich Penzel, bei Siemens für Standard-ICs verantwortlich, für den europäischen Markt gerüstet.
Auf der Basis von MIPS' Standard-RISC-Architektur wird Siemens eine weltweit unterstützte Familie von Prozessoren, Prozessorkernen und abgeleiteten Produkten anbieten. Der modulare Aufbau der Architektur macht es möglich, Kerne und andere Elemente des Prozessors für sogenannte *Embedded Controller* zu verwenden, wie sie auf dem europäischen Markt für zahlreiche industrielle Anwendungen gefragt sind.
Siemens und die weiteren Partner von MIPS stellen als einzige dem Markt RISC-Chips zur Verfügung, die volle Pin-Kompatibilität zwischen den Chips verschiedener Hersteller garantieren. *MP*



Neue Universalresiste für galvanischen Aufbau und alkalisches Ätzen

Du Pont hat eine neue Serie von Trockenfilmresisten herausgebracht, die in wäßriger Lösung verarbeitet werden und speziell für galvanischen Aufbau und alkalisches Ätzen entwickelt wurden. Die in Dicken von 38 und 50 μm lieferbaren Resiste der Riston-Serie 4600 zeichnen sich durch kurze Belichtungszeiten und einen breiten Verarbeitungsspielraum aus. Der Trockenfilmresist hat nach der UV-Belichtung einen deutlichen Farbumschlag. Die unbelichteten Teile sind hellgrün, die belichteten Teile werden dunkelgrün (unter Weißlicht dunkelblau). Dadurch erhält man einen sehr guten Kontrast zu Kupfer, was die Sichtkontrolle nach der Entwicklung erleichtert.
Der 38 μm dicke Trockenfilmresist Riston 4615 ist in den üblichen alkalischen oder sauren Ätzmedien für Print- und Ätzware einsetzbar und ist in allen in der Leiterplattenindustrie üblichen sauren Galvanobädern beständig. Der 50 μm dicke Trockenfilmresist Riston 4620 ist durch seine hohe Flexibilität und Widerstandsfähigkeit besonders gut für die Tenting-Technik geeignet.
Die neuen Riston-Trockenfilmresiste werden in 1,0prozentiger Natriumkarbonat-Lösung entwickelt und lassen sich in 1,0 bis 1,5 Prozent Kaliumhydroxid bei 45 bis 85 Grad Celsius strippen. *MP*

Supercomputer mit 64-Bit-Prozessoren

Von der amerikanischen Firma NCUBE wird seit einigen Monaten ein neuer Supercomputer angeboten. Er soll im wissenschaftlich-technischen Bereich sowie auf kommerziellem Gebiet Anwendung finden.
In dem Computer sind bis zu 8192 VLSI-Prozessoren mit einer Verarbeitungsbreite von 64 Bit integriert. Jeder Prozessor stellt ein komplettes Computersystem mit folgenden Teilen dar:
– 64-Bit-ZVE (4-Instruction-Pipeline, Daten-Cache mit 8 Registern/Operanden, Instruction-Cache mit 128 Byte)
– 64-Bit-IEEE-Gleitkommaprozessor
– Speicherverwaltungseinheit zur Fehlerkorrektur mit 32-Bit-Datenpfad
– 28 DMA-Kanäle.

Der Computer ist in der Lage, pro Sekunde 60 Milliarden Instruktionen sowie 27 Milliarden skalare Gleitkommaoperationen auszuführen. Als Benutzerschnittstellen werden Sun-3- oder Sun-4-Arbeitsplätze angeboten. So lassen sich Unix (mit ETHERNET) oder Sun-NFS als Nutzerumgebung verwenden. Der Preis des Gerätes liegt, je nach Konfiguration, bei mindestens 500 000 US-Dollar.

Quelle: Design & Elektronik vom 3. 10. 1989 Z.

Plotter mit Floppylaufwerk

Von der Firma Graphtec GmbH wurde eine neue Plotterserie für A3-Formate vorgestellt. Die Modelle MP 4300 und MP 4400 verfügen über eine 16-Bit-CPU, eine Speicherkapazität von bis zu 40 KByte sowie Stiftsortierung. In das Modell MP 4400 wurde darüber hinaus ein 3,5-Zoll-Diskettenlaufwerk integriert, das als Datenquelle für den Plotter dienen sowie Plotdaten mitprotokollieren kann. Das Laufwerk entlastet damit den Computer und erleichtert das Mehrfachplotten von Zeichnungen.

MP

Marktanteil von MS-DOS geht zurück

Einer Prognose des Marktforschungsunternehmens IDC Deutschland GmbH zufolge, soll das Jahr 1992 eine „Wende" im Auslieferungsvolumen des Betriebssystems MS-DOS bringen. Während IDC 1991 noch mit einer Auslieferung von 12,3 Mio DOS-Kopien rechnet, sollen es 1992 nur noch 11,7 Mio Einheiten sein. Die Gründe dafür sieht IDC im verstärkten Einsatz der Betriebssysteme OS/2 und Unix für PCs. 1992 sollen rund 5 % der weltweit ausgelieferten PCs unter Unix betrieben werden. 1988 betrug dieser Anteil 1,7 %. Beim Betriebssystem OD/2 prognostiziert IDC einen Marktanteil von etwa 12 % (1988: 0,4 %). Der Anteil von MS-DOS am Welt-Auslieferungsvolumen werde voraussichtlich im Zeitraum von 1988 bis 1992 von 60 auf 47 % zurückgehen.

Quelle: Eildienst. – Berlin (1989) 166. – S. 6 Fa

Bauelemente-Industrie der USA tendiert zu Multichip-Moduln

In der amerikanischen Bauelementeindustrie soll sich gegenwärtig ein neuer Trend zu sogenannten Multichip-Moduln abzeichnen. Bei diesen Multichip-Moduln handelt es sich um Prozessoren, die mit einer größeren Zahl von Speichern in einer gemeinsamen Kapsel zusammengefaßt werden. Dabei ergeben sich z. B. 32-Bit-Prozessoren mit acht DRAMs und mehr als 1 Mio Transistoren pro Modul. Die Multichip-Moduln weisen relativ geringe Abmessungen auf und werden in erster Linie als geschlossene Prozessor-/Speicher-Einheiten extremer Leistungsfähigkeit angeboten.
Die Firma Unisys verwendet bereits die neue Modultechnik in ihrem Desktop-Mainframe-Computer „A1".
Neben IBM wollen auch andere größere, mittlere und teilweise kleinere Firmen die Modultechnik einsetzen, da sie mit dieser Technik auf bewährte Einzelchips auch aus anderen Quellen zurückgreifen können und ohne höhere Integrationsstufe bei den ICs schon jetzt leistungsfähige Bauelemente erhalten. Zum anderen kann man die gleichen Prozessor-/Speicher-Kombinationen, die jetzt in Modulform erscheinen, zwar auch mit herkömmlichen ICs auf den üblichen Platinen erzeugen, aber bei dieser Modullösung sinkt die Zahl der Anschlußkontakte um 90 Prozent.
Bei einer solchen Reduktion sind innerhalb der Moduln jeweils zwei gestapelte Leiterbahnschichten erforderlich. Wollte man die gleiche Funktionsdichte pro Grundflächeneinheit auf Platinen erhalten, so wären dort (rechnerisch gesehen, aber nicht realisierbar) bis zu 30 oder 40 übereinanderliegende Leiterbahnen erforderlich.
Bei der jetzt geforderten, möglichst engen Packungsdichte der IC-Kapseln innerhalb der Moduln werden Montagesubstrate mit sehr engen Leiterbahnstrukturen erforderlich. Diese internen Verbindungen können, bei den heute verfügbaren Mitteln, mit einer Auflösung von 100 bis 500 Linien pro Zoll hergestellt werden.
An der Auffindung neuartiger Leiterbahnsubstrate für die Moduln wird gearbeitet.

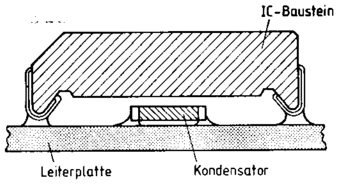
Quelle: Neue Zürcher Zeitung vom 7. 6. 1989 Fa

Chip-Strukturen aus Diamant

Die guten physikalischen Eigenschaften des Werkstoffes Diamant machen ihn für den Einsatz in der Elektronik immer interessanter. Deshalb arbeiten zahlreiche Forschungsteams an Universitäten und in Industrielabors daran, diesen Werkstoff beherrschbar und bearbeitbar zu machen. Das besondere Problem besteht darin, daß kristalliner Kohlenstoff der härteste Werkstoff überhaupt ist und bisher noch nicht, wie Silizium, als Einkristall-Kristallstab gezogen werden konnte. In einem ersten Schritt trägt man daher mikrometerdünne Kohlenstoff-Schichten auf Unterlager auf und läßt sie kristallisieren. Mehrere Forschungslabors beherrschen bereits diese Technik soweit, daß sie beginnen, die Strukturen in Form elektronischer Schaltungen aufzubringen.
Dem Forschungslabor des japanischen Stahlunternehmens Kobe Stell Ltd. gelang mittels des *reaktive-Ion-echting*-Verfahrens die photografische Übertragung von Strukturen im Bereich von 20 bis 50 μm. Als Ätzgas wird ein Gemisch aus Sauerstoff und Kohlenstoff-Tetrafluorid benutzt und unter einer Schutzgasatmosphäre von Argon gehalten. Das Unternehmen will den Prozeß in wenigen Jahren so gut beherrschen, daß es Thermistoren und Schottky-Schaltungen herstellen kann. Nach seiner Einschätzung wären dafür jedoch rund fünf Jahre Forschungs- und Entwicklungszeit notwendig.

Quelle: Blick durch die Wirtschaft vom 19. 6. 1989 Fa

Flache Keramik-SMDs



Nur 0,6 mm flach sind die neuen keramischen Chip-Kondensatoren von Siemens. Damit können sie auch platzsparend unterhalb der integrierten Schaltungen auf der Leiterplatte angebracht werden. Speziell der Chip in der Bauform 1206 mit einem Kapazitätswert von 100 nF +/− 20 % hat sich als Entkoppelkondensator für Speicherbausteine bewährt. Seine Kapazität ist für alle Speichergenerationen, also auch für den 4-MBit-Speicher, geeignet.
Der Kondensator erfüllt die Z5U-Charakteristik und ist für Reflow- und Schwallötungen mit Silber-Nickel-Zinn-Kontakten lieferbar. Für Leitklebeverbindungen steht alternativ dazu eine Silber-Palladium-Version zur Verfügung. Weitere Keramikmassen (COG, X7R) sind in verschiedenen Bauformen (0805, 1206 und 1210) erhältlich. MP

32-Bit-Mikroprozessor von Hitachi

Auf der Basis der BICMOS-Technologie entwickelte die japanische Firma Hitachi Ltd. einen leistungsfähigen 32-Bit-Mikroprozessor, der eine Verarbeitungsgeschwindigkeit von 70 Mio Befehlen pro Sekunde erreichen soll. Diese hohe Geschwindigkeit und die große Integrationsdichte von bis zu 530 000 Transistoren auf einem Chip mit einer Größe von 13 mm wurde durch die BICMOS-Technologie möglich, die die Vorteile der CMOS- mit der der Bipolar-Technologie auf einem Chip vereint. So wird eine Arbeitsfrequenz von 70 MHz bei einer Verlustleistung von nur 2,1 W erreicht. Die Strukturbreiten des neuen Prozessorchips liegen bei 1 nm.

Quelle: New Technology Japan vom Mai 1989 Fa

5400 U/min bei Plattenlaufwerk

Die Firma Imprimis (USA) arbeitet an der Entwicklung eines Plattenlaufwerkes (5,25 Zoll), das mit einer Geschwindigkeit von 5400 Umdrehungen pro Minute arbeitet. Bisher wurden diese Festplatten durchweg mit 3600 U/min betrieben. Durch die Geschwindigkeitssteigerung können die Daten nun mit höherer Frequenz (25 MHz) von der Platte gelesen werden. Außerdem soll sich die Latenzzeit auf 5,56 Millisekunden verringern. Ein neuer Aktuator hilft zusätzlich, die Suchzeit (über die volle Plattenspurbreite) auf durchschnittlich 12 ms zu senken. Das mit zehn Platten ausgestattete Gerät ist in einem Standardgehäuse untergebracht. Unbeabsichtigt kam es zu dem Nebeneffekt der hohen Drehgeschwindigkeit: Der zentrifugale Luftstrom, der die Präzision mit beeinflußt, ist gleichmäßiger als üblich.

Quelle: Elektronik. – München 38 (1989) 20. – S. 6 Fa

Transistor mit 24 „Schwellwerten"

Eine Neuentwicklung im Bereich der Quanteneffekt-Halbleiter ist ein von den AT & T Bell Laboratories vorgestellter „Multistatus"-Transistor. Mit diesem neuen Transistor sollen sich gegenwärtig bis zu 24 unterschiedliche Ein-/Ausgangspunkte für Steuersignale festlegen lassen, während es beim klassischen Transistor nur einer ist. Das würde bedeuten, daß sich in bestimmten Schaltungen bis zu 24 Einzeltransistoren ersetzen lassen. Dieser Transistor wurde bereits in Bauelementen beim Parity-Bit-Chekking, in Frequenzmultiplexern und in Konvertern mit Pegelerhöhung erfolgreich im Labor getestet.
Bei dem neuen Transistor stützt sich AT & T auf einen Quanteneffekt, der in sehr dünnen Halbleiter-Schichten von einigen Angstroem Dicke auftritt. Im Gegensatz zum normalen Elektronen- und Löscher-Transport durch die Sperrschicht, findet bei Quanteneffekt-Halbleitern eine trägerlose „Übertragung" von Energiezuständen statt. Dies hat zur Folge, daß deutlich geringere Schaltzeiten erzielt werden. Wie AT & T bekannt gab, wurde bereits eine Betriebsfrequenz von 24 GHz erreicht.

Quelle: Elektronik. – München 38 (1989) 18. – S. 7 Fa

Biochip arbeitet mit Nervenzellen

Auf dem Gebiet der Biochip-Forschung arbeiten auch die Wissenschaftler der Washington State University, unter der Leitung von Professor Dr. Bernhard Van Wie. Sie entwikkelten auf der Grundlage der Nervenzellen aus dem Gehirn von Schlangen einen Biochip, der als Sensorelement zur hochempfindlichen Registrierung von Chemikalien auch in winzigen Spuren dient. Die ersten Meßergebnisse zeigten, daß sich die bekanntlich hochentwickelte Riechfähigkeit von Schlangen nicht ausschließlich auf ihre natürlichen Beutetiere beschränkt. Die weiteren Forschungsarbeiten richten sich nun auf die Ermittlung der erfaßbaren Chemikalien. Zugleich muß die Lebensfähigkeit der Schlangen-Nervenzellen verlängert werden.
Das Forscherteam arbeitet auch mit Nervenzellen (Neuronen) anderer Tierarten, um herauszufinden, welche für langfristig operierende Sensorelemente eingesetzt werden können. Es ist aber noch nicht absehbar, ob und wann derartige Systeme industriell einsatzfähig sein werden.

Quelle: Blick durch die Wirtschaft vom 20. 6. 1989 Fa

Mikroprozessoren und Mikrorechner
Hardware und Software

von W. Meiling und R. Fülle, 1. Auflage, Akademie-Verlag Berlin, 1988, 491 S., 162 Abbildungen, 60 Tabellen, DDR 75,– M

Dieses Buch ist bereits der 25. Band der wespenfarbenen Reihe Informatik · Kybernetik · Rechentechnik. Sein Klappentext gibt als Zielgruppe „Anwender und künftige Nutzer von Mikrorechnern" an. Der erste Blick auf die vierfach unterteilte Gliederung läßt aber eher vermuten, daß dem geübten Wissenschaftler eine wohlproportionierte Gedächtnisstütze geboten werden soll. Für wen also wurde dieses Buch geschrieben?
Im Einführungskapitel werden die Begriffe Mikroprozessortechnik, Mikrorechentechnik und Mikroelektronik geklärt. Der Entwicklung der Speicher- und Prozessorschaltkreise folgt eine Zeittafel zur Entwicklung der Rechentechnik und der Mikroelektronik. Diese Informationen sind gut aufbereitet, enthalten aber für den „künftigen Nutzer" leider zu viele ungeklärte Fachbegriffe. Einige dieser Begriffe können allerdings mit dem rund 570 Stichwörtern enthaltenden Sachregister geklärt werden.
Im nächsten Kapitel werden die Rechnerarchitekturen, angefangen natürlich bei von Neumann, behandelt. Es schließt sich eine umfangreiche Behandlung des Aufbaus und der Funktion von Mikroprozessoren an.
Anschließend werden die 8-, 16- und 32-Bit-Mikroprozessoren vom i8080, Z 80 und MC 6800 bis zum i80386 und MC 68020 ausführlich mit Pinbeschreibungen und teilweise mit Befehlserläuterungen dargestellt. 52 Seiten nimmt hiervon allein die Familie i80x86 ein. Das unterstreicht die Bedeutung, die diese Prozessorfamilie zur Zeit in der Mikrorechentechnik spielt (zum Vergleich: 19 Seiten zum Z 8000 und 14 Seiten zum MC 680x0).
Das Erscheinungsdatum des Buches – Ende 1988 – läßt schlußfolgern, daß die Autoren bemüht waren, neueste Mikroprozessoren und Rechnerarchitekturen aufzunehmen. Das entschuldigt auch ein wenig die Tatsache, daß das Buch im Interesse einer schnellen Veröffentlichung als „Reproduktion" eines Rechnerausdruckes ausgeführt wurde.
Nach einem weiteren Kapitel zur Hardware, das unter anderem solche Begriffe wie Mikrocontroller, Bussysteme und Interfaces klärt, folgen rund 200 Seiten Software. Hier werden so bedeutende Betriebssysteme wie CP/M, MS-DOS, Unix und BOS 1810 in Verbindung mit einzelnen Dienstprogrammen erläutert.
Im Kapitel Anwenderprogrammierung geben die Autoren Hilfestellungen bei der Beurteilung der Qualität von Programmen; sie geben außerdem Hinweise zur Methodik der Softwareentwicklung. Der Komplex Programmiersprachen enthält unter anderem einen Vergleich so bekannter Sprachen wie Basic, Fortan, Turbo-Pascal und C. In einem gesonderten Abschnitt werden echtzeitfähige Programmiersprachen behandelt, zum Beispiel Pearl, concurrent Pascal, Modula-2, Chill und Ada.
Aufgrund der von W. Meiling und R. Fülle gewählten wissenschaftlich exakten Darstellungsweise würde ich die eingangs gestellte Frage so beantworten: Das Buch kann vor allem Studierenden, die sich mit Informatik oder Rechentechnik beschäftigen müssen, und „künftigen Nutzern von Mikrorechnern" mit einer Hoch- oder Fachschulausbildung, die sich in die Mikrorechentechnik einarbeiten wollen, empfohlen werden. *H. Hemke*

DOS 4.0 Power User's Guide

von B. Rupp, Dr. Alfred Hüthig Verlag, Heidelberg 1989, 225 S., mit Programmdiskette, ISBN 3-7785-1809-7

Ein gutes Buch, das allen Anwendern zu empfehlen ist, die beschlossen haben, für den restlichen gemeinsamen Weg mit MS-DOS die Version 4.0 zu wählen. Denn darüber besteht kein Zweifel: Angesichts des Markteintritts von OS/2 wird DOS wohl noch angepaßt und verbessert, aber nicht mehr weiterentwickelt. Überhaupt ist fraglich, ob sich diese Version zum neuen Standard mausern kann. Die gleiche Situation gab es übrigens auch bei CP/M, wo sich die letzte Version 3.0 nie so recht durchgesetzt hat. Die wirklich nennenswerten Vorteile beschränken sich auf eine grafische Nutzeroberfläche, die Verwaltung größerer Festplatten (bis 2 Gigabyte) und EMS-Unterstützung. Demgegenüber gibt es aber Inkompatibilitäten, die z. B. bewirken, daß die beliebten Norton-Utilities nicht mehr genutzt werden können. Ursache dafür ist die Erweiterung einer DOS-internen Tabelle, des DOS Parameter Blocks (DPB) im Zusammenhang mit der Verwaltung großer Festplatten. Programme, die direkt zugreifen, bleiben auf der Strecke. Diese Details wird Ihnen das Buch aber nicht verraten, denn es ist nicht für Insider gedacht. (Für derartige Probleme sei aber die c't 4/89, S. 170, empfohlen.) Der Autor nennt die Zielgruppe „mündige DOS-Anwender", was sich auch darin ausdrückt, daß das übliche Fachlatein vorausgesetzt wird. Während sich viele Begriffe einfach nicht sinnvoll übersetzen lassen, spielt aber wohl auch die amerikanische Adresse des Autors eine Rolle. Oder wissen Sie, was Troubleshooting ist?
Die Darstellung beschränkt sich nicht auf Neuerungen gegenüber der Version 3.3 – die einen gesonderten Abschnitt einnehmen –, so daß das Buch auch für Einsteiger geeignet ist. Ziel ist nicht Vollständigkeit (für die es Handbücher gibt); statt dessen wird praktisches Wissen vermittelt. Sehr gut gewählt sind die zahlreichen Beispiele, um eine Vorstellung von den erklärten Sachverhalten zu entwickeln. Dafür wurden neben der DOS-Kommandosprache auch Turbo-Pascal, Turbo-C und ANSI-Fortran verwendet, so daß jeder das neue Wissen in seiner Haussprache ausprobieren kann. Und damit der Spaß nicht bei der Tipparbeit verlorengeht, gehört eine Diskette zum Buch, die alle Beispielprogramme, die DOS 4.0-Systemfiles und eine eigene kleine Shell beinhaltet.
Breiten Raum nimmt die Installation des Systems und die Nutzung der neuen DOS-Shell ein. Das ist sicherlich sinnvoll, denn wer hat sich noch nicht über fehlende Umlaute, falsche Laufwerksformate und nicht angepaßte Drucker geärgert. Das Buch kann aber auch von Nutzern gelesen werden, die DOS 4.0 noch nicht gekauft haben, um sich über die Möglichkeiten der neuen Version zu informieren. Der Autor sagt auch, wann es keinen Sinn hat: Ohne Festplatte zum Beispiel – und auf einem 386er nimmt man besser gleich OS/2.
Positiv fällt auf, daß der Autor einerseits mit leichter Feder schreibt, auch mit persönlichen Erfahrungen und Ansichten nicht spart, andererseits die Fakten aber übersichtlich und exakt darstellt. Bei der Lektüre des Buches fühlt man sich zu keiner Zeit allein gelassen. Mit Warnungen vor zu erwartenden Widrigkeiten und Fehlern sowie mit nützlichen Hinweisen wird der Leser auf dem Weg durch das neue Terrain auf angenehme Art und Weise begleitet. Der Attacke des Autors gegen den Kopierschutz, den er „geradezu als Körperverletzung" empfindet, ist nichts hinzuzufügen. *U. Schulze*

Intel-80386-Systemprogrammierung

Übersetzung aus dem Amerikanischen von V. Sasse, Markt Technik-Verlag, Haar bei München 1989, 239 S., 78 Abb., 15 Tab., 79,– DM

Im vorliegenden Fachbuch sind zwei grundlegende Publikationen der Fa. Intel zum Mikroprozessor Intel 80386 zusammenfaßt. Der erste Teil des Buches ist eine Einführung in die Architektur (Introduction to the 80386). In kompakter, ansprechender und verständlicher Form werden die Entwurfsziele, die Funktionsprinzipien und die Struktur des 32-Bit-Mikroprozessors beschrieben. So werden im ersten Kapitel die wichtigsten Merkmale des Mikroprozessors zusammenfassend dargestellt. Im zweiten Kapitel werden Registerstruktur, Adressierung und Datentypen beschrieben. Im dritten Kapitel werden Systemregister, Multitasking-Unterstützung, dynamische Adreßumsetzung und virtueller Speicher, Schutzmechanismen, Systemruf-Organisation sowie die Interrupt-Organisation behandelt. Der vierte Abschnitt ist den Kompatibilitätsproblemen zum Intel 8086 und Intel 80286 gewidmet, und im fünften Kapitel werden die Hardwarestruktur, Pinbelegung und Busorganisation dargestellt.
Der zweite, umfangreichere, Teil des Buches ist den eigentlichen Fragen der Systemprogrammierung (80386 System Software Writer's Guide), d. h. sowohl den Fragen der systemnahen Programmierung wie auch der Implementierung von Betriebssystemen auf dem Intel 80386, gewidmet.
Im sechsten Kapitel werden die Register- und Datenstrukturen zur Tasksteuerung erklärt. Das siebente Kapitel behandelt ausführlich die Implementierung des Segmentations- und Seitensteuerverfahrens zur Realisierung des virtuellen Adreßraumes. In den Kapiteln 8 bis 11 wird auf die Programmierung von Interrupts und Exceptions, Systemaufrufen, Ein- und Ausgaberoutinen und die Initialisierung des Prozessors eingegangen. Im zwölften Kapitel werden Unterschiede zwischen den Koprozessoren 80387 und 80287 sowie Besonderheiten ihrer Programmierung behandelt. Im 13. und 14. Kapitel werden aus der Sicht der Systemprogrammierung Fragen der Kompatibilität zum 80286 und 8086 erörtert, insbesondere werden die verschiedenen Modi zur Abarbeitung von 8086-Programmen erklärt. Im letzten Abschnitt werden anschaulich Probleme der Portierung eines UNIX-System-V-Betriebssystems auf die 80386-Architektur anhand einer hypothetischen Implementation diskutiert.
Nicht enthalten sind im Buch die Intel-Schriften „80386 Programmer's Reference Manual" und „80386 Hardware Reference Manual", die für ein tiefgreifendes Verständnis und für das weitergehende Studium erforderlich wären. Auch sind die enthaltenen Programmbeispiele alle in ASM386 geschrieben, deren volles Verständnis zumindest zusätzlich eine Befehlsliste des 80386 erforderlich macht. Auch setzt der zweite Teil des Buches grundlegende Kenntnisse über Funktionsprinzipien von Multitasking-Betriebssystemen voraus.
Insgesamt kann abschließend das Fachbuch jedem interessierten Leser als die grundlegende Einführung in Architektur und Systemprogrammierung des modernen 32-Bit-Mikroprozessors Intel 80386 nur empfohlen werden. *Prof. Dr. Th. Horn*

dBase III

von W. Grafik. Berlin: Verlag Technik, 1989. 240 S., 31 Bilder, 15 Tafeln. ISBN 3-341-00703-2

Vor mehreren Monaten nun schon erschienen, ist dieses Buch ein gelungenes Lehr- und Nachschlagewerk, das sicher immer noch reißenden Absatz finden würde. Leider liegt auch hier die geplante Auflagenhöhe um Haaresbreite unter dem Bedarf, so daß sich der eine oder andere interessierte Datenbankfreak mit einem aus der Bibliothek entliehenen Exemplar helfen muß. All jenen wünsche ich, daß die zweite Auflage noch vor dem weltweiten Übergang von dBase III zu dBase IV erscheint. Einige Beratungsmuster sollen ja auch schon vorhanden sein.
Die dBase-Nutzer ohne spezielle Vorkenntnisse sind mit diesem Buch besonders angesprochen; wer schon Erfahrungen mit dBase II hat, kann einige Abschnitte überfliegen und wird es wahrscheinlich mehr zum Nachschlagen nutzen. Ein- und Umsteiger kommen so gleichermaßen auf ihre Kosten, insbesondere auch durch die in Ergänzung zu der Referenzkarte in der MP 3/89 sehr ausführliche Beschreibung aller dBase III-Kommandos mit vielen Querverweisen zu ähnlichen oder zusammengehörenden Kommandos und zu den Abschnitten des Buches, in denen sie ausführlich beschrieben sind.

Auch die Ergänzungen in dBase III Plus sind als letzte Aktualisierung im Anhang enthalten.
Nach einer Einführung in die Grundlagen der dBase III-Datenbankarbeit mit vielen wichtigen Informationen, die zudem leichtverständlich und doch exakt gegeben werden, folgt – meiner Ansicht nach ein wenig früh die *Besichtigung einer Datenbank.* Man könnte jetzt streiten, wer zuerst auf dem PC war, dBase oder die Datenbank. Ich meine aber, daß das Kommando CREATE zum Anlegen einer Datenbank auf Seite 68 vielleicht doch etwas weit hinten liegt. Vielleicht aber gibt es bei dBase auch so etwas wie unbefleckte Empfängnis. Nun gut, wie heißt es, ein Blick ins Buch und zwei ins Leben. Und so wird der findige Leser schon ergründen, wo es das gibt, was er braucht. Die Benutzung der Datenbank wird dann aber wie schon gewohnt genau, gründlich und mit vielen Beispielen erläutert, die jeder (wenn er erst mal eine hat) leicht nachvollziehen kann. Interessant dürften auch die vielen kleinen Tips und Tricks sein, die wie nebenbei mitgeliefert werden. So kann man beispielsweise ab der deutschen Version 1.1 davon ausgehen, daß die Umlaute und der Buchstabe β richtig einsortiert werden. Vielfältige Möglichkeiten bieten auch die gegenüber dBase II neuen, ausführlich beschriebenen Datumsfunktionen und die Hinweise zur direkten softwaremäßigen Steuerung des Druckers mit dBase. Die Beispiele sind oft durch den direkten Abdruck des Bildschirminhalts aufgelockert und ergänzt, so daß Sie sofort überprüfen können, ob Sie das gleiche Ergebnis auf Ihrem PC erreichen. Der Autor macht auch auf Fehlerquellen aufmerksam, die beispielsweise beim Anhängen (APPEND) von Datensätzen einer Datei mit anderer Struktur auftreten können. Wenn die Felder der Zieldatei beispielsweise kürzer sind als die der Quelldatei, werden die restlichen Zeichen ohne Fehlermeldung abgeschnitten. Besonders ärgerlich, wenn man es nicht gleich merkt. Praktische Beispiele verdeutlichen auch das Aufbereiten und Ausdrucken von Listen für den schnellen aktuellen Überblick. Ganz ohne Statistik geht's nun mal nicht.
Ein vergleichsweise leider etwas geringer Teil (14 Seiten) wird direkt der Programmierung mit dBase III gewidmet; die prinzipielle Arbeit mit Kommandodateien aber an Beispielen nachvollziehbar erläutert, ebenso die Arbeit mit Prozedurdateien und die Nutzung von Parametern.
Mir gefällt an diesem Buch besonders, daß nicht alles bejubelt wird, sondern sachlich die riesigen Möglichkeiten von dBase III den Grenzen und Mängeln gegenübergestellt werden. So ist es Ashton-Tate immer noch nicht gelungen, einen akzeptablen Texteditor für MODIFY COMMAND in das System zu integrieren; eine kleine Anleihe bei Borland könnte da vielleicht Abhilfe schaffen. Aber das Installieren von dBase III durch die SET-Kommandos oder das Ändern der Datei CONFIG.DB kann nicht nur diesen, sondern auch andere Mängel beheben. 26 Seiten sind der kompletten, alphabetischen Befehlsübersicht gewidmet; im Anhang sind alle Funktionen von dBase III aufgeführt, und es wird auch ein Ausblick auf dBase III Plus gegeben.
Alles in allem ein Buch, das ein dringendes Informationsbedürfnis unserer PC-Nutzer befriedigen kann.

J. Hill

Bussysteme in Mikrorechnerautomatisierungsanlagen

von J. Sawatzky. Schriftenreihe der Betriebssektion der KDT des VEB GRW „Wilhelm Pieck" Teltow 1986, Heft 16, 86 S., 48 Abb., 7 Taf., 64 Lit., DDR 8,– M

Der Autor legt mit dem Heft 16 eine längst fällige Monographie zur Problematik der busgebundenen Datenübertragung vor. Dezentral strukturierte mikrorechnergestützte Automatisierungsanlagen basieren auf der digitalen Informationsverarbeitung, -übertragung und -speicherung. Die digitale serielle Informationsübertragung wird in derartigen Strukturen durch Bussysteme realisiert. In der Broschüre geht der Autor ein auf
– Stellung und Bedeutung der seriellen Datenübertragung in Automatisierungsanlagen
– Strukturen und Funktionen der Bussysteme
– das Schichtenmodell der Datenübertragung
– Standardisierung von Datenübertragungssystemen
– ausgewählte Anlagenbussysteme sowie
– Entwicklungstendenzen bei Anlagenbussystemen.
Ausgehend von dem im Prozeßleitsystem audatec verwendeten audatec-Bussystem, werden die Parameter leistungsfähiger Anlagenbussysteme ausgewählter internationaler Prozeßleitsysteme miteinander verglichen. Auf Notwendigkeit, Stand und Perspektive der Standardisierung von digital-seriellen Datenübertragungssystemen wird eingegangen. Der Autor schätzt ein, daß neben den weit verbreiteten Anlagenbussystemen die Feldbussysteme zunehmend an Bedeutung gewinnen werden. Abgerundet wird die Broschüre durch ein Abkürzungs-, Bild- und Tafelverzeichnis sowie ein umfangreiches Literaturverzeichnis, das dem Autor ein tiefgründiges Quellenstudium bescheinigt. Ein Sachwortverzeichnis mit der Möglichkeit des kurzfristigen Einstiegs in die anspruchsvolle Thematik wird vermißt. Der Problemkreis der digital-seriellen Datenübertragung wird so verständlich und übersichtlich dargestellt, daß sie einen breiten Leserkreis anspricht.
Zu beziehen ist die Veröffentlichung über die Betriebssektion der KDT oder die Zentrale Informationsstelle Wissenschaft und Technik des VEB GRW „Wilhelm Pieck" Teltow, Oderstraße 74–76, Teltow, 1530, zum Preis von 8,– M.

Blackert

C für Profis

Techniken – Anwendungen – Konzepte

von D. E. Krekelberg und G. E. Sobelman. Übersetzung aus dem Amerikanischen von A. Stocklauser Markt Technik-Verlag, Haar b. München, 1987, 293 S., 47 Abb., 4 Anlagen, 1 Diskette, 69,– DM

Das vorliegende Fachbuch zweier namhafter amerikanischer Spezialisten wendet sich in erster Linie an „professionelle" C-Programmierer. Die Kenntnis des Sprachstandards gemäß B.W. Kernighan und D.M. Ritchie „The C Programming Language" oder vergleichbarer Publikationen und praktische Programmiererfahrungen werden grundsätzlich vorausgesetzt.
Im ersten Kapitel werden einige grundlegende Einführungen zum Top-Down-Programmentwurf, zur Programmstruktur und Programmportabilität gemacht. Im zweiten Kapitel werden als weitere Grundlage für die nachfolgenden Kapitel ausgewählte Probleme aus dem Themenkreis Zeiger und Felder sowie die Funktionen zur dynamischen Speicherverwaltung behandelt. Das dritte Kapitel gibt eine Einführung in Strukturen, Vereinigungen (Varianten) und Bitfelder, und das vierte Kapitel behandelt speziell den Themenkreis rekursiver Funktionen im Vergleich zu Iterationen und den Funktionsaufruf über Funktionszeiger. Die Rekursion wird sehr anschaulich am Denkspiel „Turm von Hanoi" dargestellt.
Auf der Grundlage dieser Kapitel wird im zweiten Teil des Buches mit den Kapiteln 5 und 6 eine hervorragende Einführung in Listen und Bäume gegeben, die in der professionellen Programmierung von der Datenverwaltung bis hin zur Compilertechnik sehr weit verbreitet sind. Bei den Listen werden einfach und doppelt gekettete Listen, Stapel, FIFO-Schlangen und Listen von Listen behandelt. Bei den hierarchisch strukturierten Bäumen werden vor allem die Auswertungsstrategien und Baumoperationen diskutiert und an Beispielen gezeigt.
Im abschließenden dritten Teil des Buches wird, aufbauend auf den beiden ersten Teilen, ein anspruchsvolles professionelles Grafikinterface erarbeitet, das auf IBM-PCs mit Farbgrafik-Adapter unter MS-DOS und anderen Computern mit pixelfähigen Bildschirmen unter UNIX eingesetzt werden kann. So wird im Kapitel 7 eine Einführung in Grafiksysteme gegeben, ohne dabei auf internationale Standards wie GKS einzugehen, deren Implementierung für PCs zu aufwendig ist. Dieses Kapitel dient der Einführung in ein segmentorientiertes Grafiksystem, dessen vollständiger Quellkode im Anhang A abgedruckt ist. Die Kapitel 8 und 9 beschreiben eine anspruchsvolle Benutzerschnittstelle mit Fenstern und Menüs, ähnlich wie es vom Apple Macintosh oder von Atari-ST-Rechnern her bekannt ist. Die vollständige Funktionsbibliothek zur Realisierung dieser Benutzerschnittstelle beinhaltet Anlage B.
Die beiliegende Diskette enthält den vollständigen Quellkode beider Funktionsbibliotheken, die Grafik-Primitiven zum Implementieren der Funktionsbibliotheken auf einem IBM-PC und Demonstrationsprogramme.
Abschließend sei dieses Fachbuch allen Lesern empfohlen, die sich vertiefende C-Kenntnisse aneignen wollen oder sich speziell für die Programmierung von Listen und Bäumen bzw. für die Implementierung von anspruchsvollen Grafikschnittstellen interessieren.

Prof. Dr. Th. Horn

Rationell auf dem PC programmieren

von G. Wostrack, Dr. Alfred Hüthig, Verlag, Heidelberg 1989, 296 Seiten, ISBN 3-7785-1687-6
Die in diesem Buch behandelten grundverschiedenen Sachgebiete werfen die Frage nach der Zielgruppe auf. Mir fallen da nur Informatikstudenten ein und vielseitig interessierte – aber theoretisch vorbelastete – Freaks: Zu breit gefächert ist das Themenspektrum.
Fraglich ist doch, ob sich der Programmierer einer Inferenzmaschine in Prolog auch für speicherresidente Programme in Assembler interessiert und ob gleichermaßen Hilfe für die Syntaxanalyse von Programmiersprachen und die Berechnung verdeckter Kanten grafischer Objekte vonnöten ist. Womit einige Themen schon genannt wären. Nach jedem Kapitel fragt man sich: Schon alles? und wünscht sich ein ganzes Buch darüber. Denn die Darstellung der recht komplexen Sachverhalte erfolgt so, wie man es sich öfter von Büchern wünschte: Übersichtlich und verständlich. So wird dem Leser für das Umlenken der Tastatureingabe auf eine eigene Routine nicht nur ein Programmrahmen angeboten (wie man ihn auch anderswo findet), sondern in einem Bild der genaue Weg eines Zeichens von der Hardware in den Tastaturpuffer gezeigt. Auf gleiche Weise werden Breiten- und Tiefensuchverfahren in Binaerbäumen und vieles mehr erläutert – alles Sachverhalte, deren Erschließung aus einer profanen Beschreibung heraus sich schwierig gestaltet. Das macht den eigentlichen Wert des Buches aus.
Der Autor will das Buch aber auch als Sammlung nützlicher Routinen für stets wiederkehrende Aufgaben verstanden wissen. So offeriert ein Kapitel eine vollständige (und reichlich kommentierte) Prozedursammlung für Pull-Down-Menüs, die hoffentlich viele Leser zur Gestaltung einer zeitgemäßen Nutzeroberfläche für ihre Programme anregt. Die Nutzung vorgefertigter Module ist dafür unumgänglich, wenn nicht mehr Zeit für das Programm-Outfit verwendet werden soll als für dessen eigentlichen Inhalt. Wer die Maus nutzen will, findet auch gleich noch eine Schnittstellenbeschreibung, die man oftmals vergeblich sucht. Das Beispiel für natürlichsprachlichen Dialog ist wohl eher Spielerei. Ernsthaft und fesselnd dagegen sind die Ausführungen zu Strategiespielen. Vorgestellt wird das Amazonenspiel, bei dem abwechselnd Figuren positioniert werden, die sich wie Dame und Springer beim Schach bedrohen. Eine Gewinnstrategie als Pascal-Programm steht für das prinzipielle Vorgehen bei Strategiespielen.
Einige Themen können leider nur als Anregung verstanden werden – so die Arbeitsweise von Programmen zur Rechtschreibprüfung auf 6 Seiten.
Die Hälfte des Buches nehmen Beispiele ein, teils nur den Weg andeutend, teils vollständig ausformuliert. Damit sollen jedoch keine fertigen Lösungen für die aufgeworfenen Probleme geliefert werden. Vielmehr dienen sie dazu, allgemeine Prinzipien moderner Softwareentwicklung für den Entwurf strukturierter, wartungsfreundlicher und weitgehend verifizierbarer Programme zu vermitteln. Dazu verlangt es vom Programmierer aber etwas Disziplin. Wie sagt doch der Autor: „An sich sind die Grundsätze des Software-Engineering so einfach und überzeugend, daß jeder sie fast automatisch anwenden könnte."

U. Schulze

Für Forth-Interessierte:

Von der Interessengemeinschaft Forth der Kammer der Technik in Ilmenau wird auf folgende neue Broschüre der KDT hingewiesen:
Krapp, M.; Schwartz, J.; Jahn, D.:
Handbuch zu singleFort, scpFORTH, multiFORTH

Der Vertrieb erfolgt über den KDT-Bezirksverband Suhl, PSF 510, Suhl, 6000.

Entwicklungssystem für U 88x-Programme

Das Programmpaket **GEMMA** ist ein vollständiges Entwicklungssystem mit integriertem Editor, Assembler, Reassembler, Debugger, Simulator und mit zwei universellen Emulator-Schnittstellen. Das Entwicklungssystem ist für IBM-PCs geschrieben und auf den Modellen XT und AT sowie auf Kompatiblen lauffähig. Vorausgesetzt werden mindestens 350 KByte RAM, das Betriebssystem MS-DOS als Version 3.0 und ein beliebiger Video-Adapter, der 80 Zeichen pro Zeile darstellen kann. Die Bedienerführung des integrierten Systems erfolgt mittels Pull-down-Menüs und ermöglicht damit eine unkomplizierte und gebräuchliche Benutzung (analog zu Turbo-Pascal 5.0). Helps sind in jeder Ebene aufrufbar.
Bei Interesse kann eine Demo-Version angefordert werden.

Humboldt-Universität zu Berlin, Sektion Elektronik, Bereich Technische Informatik, Invalidenstraße 110, Berlin, 1040; Tel. 2 80 35 75 *Prof. Dr. Zaremba*

P8000-Gerätetreibersoftware unter WEGA

Es werden Gerätetreibermodule für folgende periphere Geräte zur Nachnutzung angeboten:
- Plotter K 6418
- Seriendrucker SD 1152
- Schreibmaschine S 6120.

Zum Lieferumfang gehören der entsprechende Treibermodul sowie eine Anleitung für das Einbinden des Treibers in den Systemkern.

VEB Nachrichtenelektronik Greifswald, Abt. H/HE, Koll. Philipp, Brandteichstraße 25, Greifswald, 2200; Tel. 6 62 93 *Rose/Schidlowski*

Entwicklungswerkzeuge für comFORTH 2

Zur Unterstützung der Softwareentwicklung unter dem System comFORTH 2 wurde eine Reihe von Werkzeugen entwickelt, die sowohl die Entwicklung großer Programme unterstützen als auch den Programmierer von Routineaufgaben entlasten. Zu den unter der Bezeichnung **Entwicklungspaket** angebotenen, nicht betriebssystemabhängigen Komponenten gehören:
- leistungsfähiger Bibliothekar zur Verwaltung großer Quelltextbestände
- mächtige Zeichenkettenbefehle (lexikalische Vergleiche und Suchbefehle)
- Wortschatz zum Zugriff auf unstrukturierte (Wordstar-)Dateien (Lesen und Schreiben von Zeichen, Blöcken und Zeilen, Laden/Listen von Textdateien)
- automatische Generierung von Glossarrahmen während des Ladevorgangs, Sortieren und Verbinden von Glossardateien
- konfigurierbare On-line-Dokumentation – Umlenkung der Ein-Ausgabekanäle (SUBMIT, Ausgabe auf Datei, Parallelschaltung von Ausgabegeräten, Anfertigen von Sitzungsberichten)
- komfortabler Druck von Quellprogrammen.

Bitte fordern Sie weiteres Informationsmaterial an.

Wilhelm-Pieck-Universität Rostock, Sektion Technische Elektronik, WB Automatische Steuerungen, Albert-Einstein-Straße 2, Rostock, 2500; Tel. 40 52 02 *Dr. Pfüller*

MAK-Werte-Speicher

Dieser **rechnergestützte Datenspeicher ist** als Hilfsmittel für Sicherheitsinspektoren, WAO-Mitarbeiter, Projektanten und Leiter konzipiert. Er ermöglicht eine schnelle Information zu 500 Substanzen aus dem Giftgesetz und aus der MAK-Werte-Liste (TGL 32610/02), bezüglich wesentlicher arbeitshygienischer, arbeitsmedizinischer, brandschutztechnischer und physikalischer Daten sowie spezieller Hinweise zur Verwendung der Substanz. Außerdem wird auf insgesamt 9 weitere Informationsquellen mit Standortangabe, Inhalt und Nutzerbedingungen hingewiesen, wobei jeder Substanz nur die Quellen zugeordnet sind, die auch Angaben zur betreffenden Substanz enthalten.
Die Nutzung erfolgt in einem einfach handhabbaren Bildschirmdialog. Es ist eine gezielte Recherche mit verschiedenen Suchbegriffen (Namen, Namensfragmente, CAS-Nummer, Satznummer) möglich, in deren Ergebnis man alle Informationen zu einer Substanz (auch gedruckt) erhalten kann.
Es schließt sich eine Recherche nach den genauen Bezeichnungen, Inhalten, Standorten und Nutzungsbedingungen der zutreffenden Quellen an. Weiterhin ist es möglich, Kennblätter einzelner Substanzen sowie Listen der Quellen, von deren Standorten und Listen mit allen Substanzen ausdrucken zu lassen. Eine Anwenderdokumentation mit ausführlicher Begriffserklärung gestattet eine rasche Einarbeitung.
Zur Nachnutzung ist ein Rechner vom Typ A 7100 mit dem Betriebssystem SCP 1700 nötig. Das Programm befindet sich mit der Basissoftware REDABAS auf einer Diskette (5,25", DD/DS, SCP-Format, 624 KByte) und benötigt darauf einen Speicherplatz von 300 KByte. Der Bezug ist ab sofort möglich. Es ist ein Änderungsdienst vorgesehen.

Zentralinstitut für Arbeitsschutz beim Staatssekretariat für Arbeit und Löhne, Forschungsgruppe Technischer Schadstoffschutz, Gerhart-Hauptmann-Straße 1, Dresden, 8020, Tel. 47 05 72 *Dr. Röder/Dr. Bux*

CP/M 2.2+ auf MC 80.3x

Die Geräteserie MC 80.30 (31, 33) verfügt über gute Hardwarevoraussetzungen, um als grafikfähiges Steuersystem (K 1520-Bus), Entwicklungssystem oder Bürocomputer verwendet zu werden. Durch die Verwendung eines Kassettenlaufwerkes als Massenspeicher wird das allerdings stark eingeschränkt. Um die gegebene Hardware voll ausnutzen zu können und Kompatibilität zu CP/M zu erzielen, stand die Aufgabe, Floppy-Disk-Laufwerke anzuschließen und ein entsprechendes Betriebssystem zu installieren. Bedingungen dabei waren: geringer Aufwand und minimale Veränderungen des Rechners. Diese Aufgabe wird durch den Einsatz einer Floppydisk-Ansteuerleiterplatte auf der Basis des Schaltkreises U 8272 und der Nutzung des Betriebssystems CP/M 2.2+ gelöst. Um die notwendigen 64 KByte RAM zu erhalten, wird zusätzlich eine 16-KByte-DRAM-Speicherkarte verwendet. Die anzuschließenden Floppydisk-Laufwerke (maximal 4 Stück) werden beigestellt.
Durch die Umrüstung können Hochsprachen und Datenbanksysteme wie dBase, Turbo-Pascal, Forth und C verwendet werden. Der MC 80.3x kann weiterhin unter seinem Original-Betriebssystem arbeiten (durch den Start eines COM-Files). Die Original-Hardware (Kassettenmagnetband, EPROM-Programmer) ist erreichbar. Der Anschluß an einen P8000 als Remote-Terminal ist problemlos möglich. 5¼"-Laufwerke (1.2, 1.4 oder 1.6) und 8"-Laufwerke (MF 3200, MF 6400, EC 5074) können wahlfrei oder gemischt angeschlossen werden.
Es stehen durchkontaktierte Leiterplatten und Dokumentationen für die Floppy-Disk-Ansteuerung zur Verfügung.

Technische Hochschule Ilmenau, Sektion Gerätetechnik, PSF 327, Ilmenau, 6300; Tel. 7 40 *Mangold*

TURBO-Assistent

Der TURBO-Assistent ist ein Programm, welches im Dialog mit dem Bediener selbständig Programme entwickelt. Er ist 52 KByte groß, belegt 4 Dateien und ist auf allen Turbo-Versionen ab 3.0 lauffähig. Seine Entwicklung basiert auf drei Aspekten.
① Aspekt für Einsteiger: Um Computer schnell einsetzen zu können, werden höhere Programmiersprachen, z. B. Turbo-Pascal, wegen der Programmierdauer verworfen. Aber nach gewisser Zeit können Programmerweiterungen nicht oder nur mit noch höherem Aufwand durchgeführt werden. Hier bietet der TURBO-Assistent die Möglichkeit, Soforterfolge, erweiterbare Programmodule und ordentliche Programmiersprache zu vereinen. Ein komplettes Programm zur Dateiarbeit kann in nur 10 min programmiert, übersetzt und im Fachbereich angewendet werden.
② Aspekt für Programmierer: Der Turbo-Pascal-Programmierer verwendet immer wieder die gleichen modifizierten Steuer- und Unterprogramme zur Bearbeitung neuer Probleme. Äußerst hilfreich ist dabei ein Programmgerüst, auf daß er das neue Programm aufbauen kann. Die vom TURBO-Assistenten erzeugten Quellprogramme sind zu 90 % deckungsgleich. Der Programmierer greift immer wieder auf das gleiche Programmgerüst zurück.
③ Aspekt für Softwareabteilungen: Programme unterliegen der Programmpflege bzw. der Softwarewartung. Durch die breite Anwendung des TURBO-Assistenten haben alle Programme das gleiche, gut kommentierte Grundgerüst und gleiche Unterprogramme. Das erleichtert die Softwarewartung durch andere Programmierer erheblich.

VEB Wärmegeräte und Armaturenwerk Berlin, ODV, Seelenbinderstraße 129, Berlin, 1170; Tel. 6 50 90 *Dähne*

Datenauswertesystem UNIDAT

Eine erhebliche Einsparung von Programmierkapazität wird durch das universelle Datenauswertesystem UNIDAT erreicht. UNIDAT ist ein unter dem Betriebssystem DCP lauffähiges Programmpaket zur universellen Auswertung von REDABAS-Datenbanken; eine SCP-(8-Bit-)Variante in reduzierter Form liegt ebenfalls vor. Die drei Hauptbestandteile von UNIDAT sind:
1. Beliebige Verknüpfung von 2 unterschiedlichen Datenbankdateien zu einer Auswertedatei über 1 bis 3 Zuordnungskriterien
2. Auswerten einer beliebigen Datei über Drucklisten oder Zählberichte (Statistiken)
3. Druck von Karteikarten, Rechnungen und dergleichen aus dem Dateninhalt einer Datei.

Das gesamte Programmsystem arbeitet konsequent menügeführt und erfordert keine Kenntnisse der REDABAS-Programmierung.
Für das gesamte System existiert eine Kurzdokumentation.

VEB Leuna-Werke „Walter Ulbricht", Direktion Kader und Bildung, Sektor Kadererfassung, Leuna, 4220; Tel. Merseburg 4 30 *Schüler*

Unterstützung für REDABAS-4-Anwendung

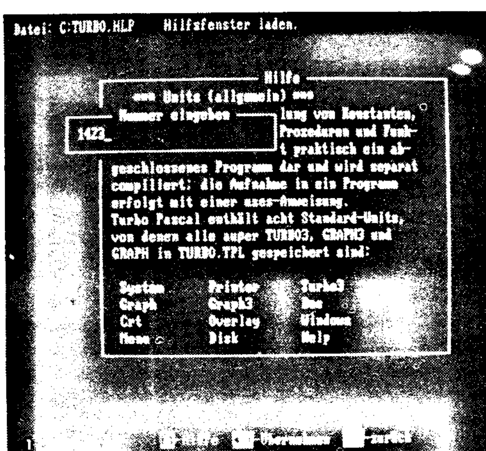
Zur Verbesserung der Anwendung von REDABAS-4 entstanden in unserem Betrieb Module, die unter Steuerung des Interpreters oder als Module für den REDABAS-Compiler (REDACOM) abgearbeitet werden können. Folgende Funktionen werden zur Zeit realisiert:
Fenstertechnik: Beliebige Fenster des REDABAS-Bildschirmes können in den Hauptspeicher gerettet und an gleicher oder anderer Bildschirmstelle wieder hergestellt werden. Beliebige Fenster können über den Drucker ausgegeben werden.
Kursorgröße: Der Kursor kann stufenweise von unsichtbar bis zu maximaler Größe eingestellt werden.
Eine Weiterentwicklung der Module ist in Arbeit.

VEB Datenverarbeitungszentrum Halle, Abteilung FP, Block 081, Halle-Neustadt, 4090 *Gruhl*

32. Zentrale Messe der Meister von morgen

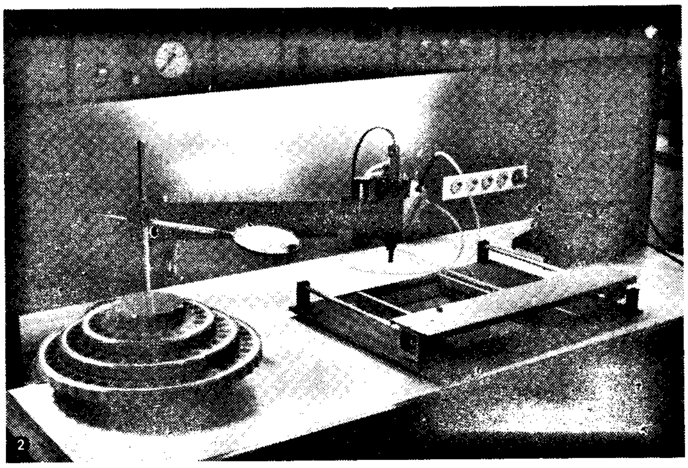
Vom 6. bis 13. November 1989 fand in Leipzig die 32. ZMMM statt. Wir haben für Sie einige interessante Exponate ausgewählt.

Professionelle Anwendersoftware kann mit dem jetzt in neuer Version vorgestellten **Window-Development-System** (WDS) effektiv programmiert werden. Mit diesem in Gemeinschaftsarbeit von Softwarearbeitsgemeinschaften der Hochschule für Ökonomie und der Humboldt-Universität Berlin entwickelten *Werkzeugsystem* läßt sich anpassungsfähige Software programmieren, die jederzeit schnell und in hoher Qualität aktualisiert werden kann (siehe **Farbbild 1**; alle Farbbilder auf der 3. Umschlagseite). Im Dialog wird die Menüführung eines Softwareprodukts festgelegt und daraus abgeleitet generiert das System WDS automatisch den Quelltext für Turbo-Pascal ab Version 4.0 einschließlich des Aufrufs der einzelnen Prozeduren nach der entworfenen Menüfolge. Dem Softwareentwickler bleibt nur noch seine eigentliche Arbeit: das Programmieren bzw. das Einkopieren des Quelltextes der Prozeduren zur Eingabe, Berechnung, Anzeige oder zum Ausdruck der im Programm zu verarbeitenden Daten. Mehrere Units – so zur skalaren Eingabe, mit Routinen zur Plattenmanipulation und zur Verwaltung von Datum und Uhrzeit – werden ebenfalls angeboten, um dem Programmierer auch noch das Schreiben von Standardprozeduren abzunehmen. Neben den menüorientierten Programmstrukturen können aber auch die farbliche Gestaltung und ein leistungsfähiges Hilfssystem im Dialog programmiert werden.
Dazu dient insbesondere **das Help-Development-System.** Es wird dann automatisch der entsprechende Quelltext generiert, der zur Einbindung in das Softwareprodukt erforderlich ist. Dieser Turbo-Pascal-Quelltext erinnert bei seiner Aktivierung im Programm nicht nur vom äußeren Erscheinungsbild an die vom Turbo-Pascal-System selbst gegebene On-line-Hilfe, sondern auch die innere Struktur ist identisch, so daß der Nutzer von HDS auch die Hilfstexte des Turbo-Pascal-Systems selbst den persönlichen Wünschen anpassen kann, und auch die Ergänzung der Texte beispielsweise um eigene Units ist möglich (Bild 1). Aus derselben Zusammenarbeit von HfÖ und Humboldt-Universität rührt ebenfalls das **Maschinentagebuch** her, das neben der Verwaltung von mehreren Standard- und eigenen Projekten für 20 Nutzer auch einen Paßwortschutz und die Auswertung des Rechnereinsatzes sowohl nach Nutzern als auch nach Projekten bietet (Farbbild 2). Dieses Paket ist als Shareware (kostenlos) gegen Einsendung einer 5¹/₄-Zoll-Diskette zu erhalten.
(Klub der Wissenschaften im Magnushaus, AG Software, Am Kupfergraben 7, Berlin, 1080)



Der integrierte Schaltkreis **U 80617** (Farbbild 3) wurde vom VEB Robotron-Elektronik Dresden gezeigt. Er gehört zum 32-Bit-Mikroprozessorsystem U 80700. Gemeinsam mit dem U 80614 und dem U 80619 wird er den Rolanet 2-Controller für den K 1820 von Robotron bilden. Dieser Controller entspricht der Ethernet- und der IEE 602.3-Spezifikation. Der U 80617 dient dem Anschluß dieses Controllers an den MPI-Bus (SKR). Er ist ein Register- und Zählerschaltkreis für die Steuerung des schnellen Datenaustausches mit direktem Zugriff auf den Hauptspeicher (DMA). Der in einer 1,5-μm-CMOS-Technologie gefertigte Schaltkreis besitzt TTL- und CMOS-kompatible Ein- und Ausgänge. Er befindet sich in einem 68poligen LCC-Keramikgehäuse (im Farbbild 3 unverkappt).
(VEB Robotron-Elektronik Dresden)

Eine wesentliche Voraussetzung für die routinemäßige Verarbeitung von SMD-Bauelementen unter den Bedingungen eines Elektroniklabors stellt das Vorhandensein eines geeigneten, stationären Arbeitsplatzes dar. Der **SMD-Handbestückungsplatz** unterstützt die Arbeitsgänge Bestükken, Löten, visuelle Kontrolle und Reparatur von gemischt bestückten Leiterplatten (Bild 2). Er umfaßt folgende Komponenten:
– verstellbare Halterung für Leiterplatten unterschiedlicher Größe mit verschiebbarer Handauflage
– Karussellmagazin für lose SMD-Bauelemente mit direktem Zugriff (im Bild links)
– verschiedene Saugpipetten mit zentraler Pumpe

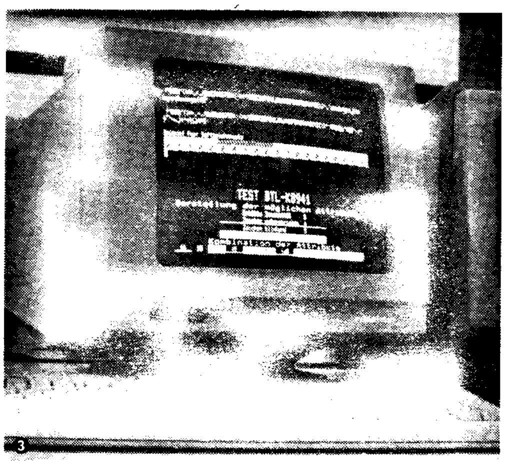

2

– Heißluftauslötvorrichtung mit Saugpipette
– Beleuchtungseinrichtung mit Großfeldlupe
– angepaßtes Werkzeug
– Lager für SMD-Bauelemente.
Die maximal mögliche Leiterplattengröße ist 130×190 mm²; der Arbeitsplatz hat die Abmessungen 1,35×0,7×1,7 m³.
(Martin-Luther-Universität Halle, Sektion Physik, Dr. Rost/Balzerowski, Friedemann-Bach-Platz 6, Halle, 4010; Tel. 2 68 20)

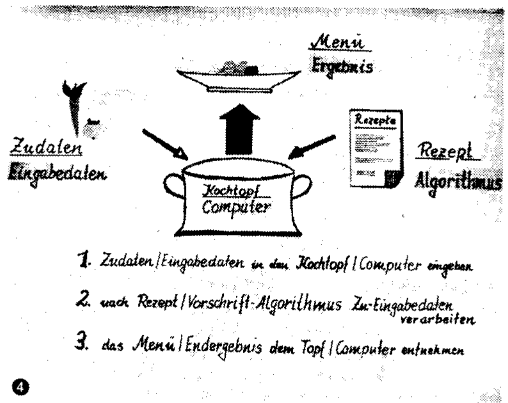
Die im Farbbild 4 dargestellte **Großfeldtastatur** ist eine Entwicklung der Sektion Rehabilitationspädagogik und Kommunikationswissenschaft der Humboldt-Universität. Sie ist eine Spezialtastatur und für den Gebrauch durch Körperbehinderte und andere Geschädigte im schulischen und außerschulischen sowie im privaten Bereich vorgesehen. Diese Tastatur kann mit einem Anschlußmodul an den KC 87 angeschlossen werden. Über diesen Anschlußmodul ist später auch die Kopplung an andere Computer (z. B. Bildungscomputer) möglich.
Zum Betreiben der Tastatur wird des weiteren noch ein spezieller Druckermodul, ein spezieller ROM-Erweiterungsmodul mit dem Textprogramm RETEXT, ein RAM-Erweiterungsmodul sowie ein spezieller Kodierstecker benötigt. RETEXT gestattet dem Nutzer, Schreibarbeiten und mathematische Aufgaben rationell zu erledigen. Die Buchstaben auf der Tastatur sind ergonomisch angeordnet, um die Fingerwege so gering wie möglich zu halten. Die Häufigkeit der Buchstaben, Buchstabenfolge, der Wörter und der Anfangsbuchstaben der deutschen Sprache bildeten die Grundlage für die Buchstabenanordnung. Es konnten dadurch die Fingerwege auf 55 % und die Schreibzeit auf 60 % gesenkt werden. Die halbkreisförmige Anordnung der Tasten ist ebenfalls ergonomisch (Grundlage dafür ist der antropometrische Abstand zwischen Ellenbogen und Fingerspitze); damit ergeben sich minimale Betätigungswege bei fester Auflage des Ellenbogens.
Der Tastentiefgang kann zur Anpassung an die individuellen Bedürfnisse des Nutzers in fünf Stufen eingestellt werden, und die Neigung der Tastatur zur Tischfläche ist von 0 bis 65 Grad stufenlos möglich. Eine kompakte und schwergewichtige Bauweise und Gummifüße garantieren eine rutschfeste und sichere Aufstellung. Hersteller ist der VEB Robotron-Elektronik Dresden.
(Humboldt-Universität zu Berlin, Sektion Rehabilitationspädagogik und Kommunikationswissenschaft, Dr. Kurt/Golomb, Albrechtstraße 22, Berlin, 1040; Tel. 2 80 51 87)

Bereits in unserem Heft 10/89 kündigten wir in unserem Bericht vom 13. Mikroelektronik-Bauelementesymposium (Mai 1989 in Frankfurt [Oder]) die Turbo-Karte für den EC 1834 – die **BK 600** – an. Erstmals in Funktion zu sehen war sie aber erst zur letzten ZMMM (siehe Farbbild 5). Hier noch einmal die wichtigsten technischen Daten: CPU U 80601-8, Buscontroller U 80606-8, Taktgenerator U 80612-8, Koprozessor U 80613 (im Bild der leere Steckplatz), 1 MByte RAM mit 256-KBit-RAM-Schaltkreisen U 61256 (150 ns), SRAM U 214/6514.
Die Karte, die in jedem EC 1834 verwendet werden kann, ermöglicht eine Taktrate von 8 MHz ohne Wartezyklen der CPU (0 Waitstates). Damit kann der EC 1834 auf die 3- bis 4fache Rechengeschwindigkeit beschleunigt werden (mit U 80613: 5- bis 8fach). Die lange Steckkarte besteht aus 4 Leiterzugebenen und wird im Schwierigkeitsgrad 6 gefertigt.
(VEB Mikroelektronik „Karl-Marx" Erfurt, Forschungszentrum, Koll. Bankel, Rudolfstraße 47, Erfurt, 5010; Tel. 41 20 13, App. 79)

Vom VEB Robotron-Elektronik Zella-Mehlis wurde das alphanumerische Terminal **K 8941** ausgestellt (Bild 3). Die Elektronik ist im Monitorgehäuse enthalten, beispielsweise ein U 884, ein Bildschirmcontroller und 30 KByte Nutzersoftware im ROM. Das Terminal ist kompatibel zum VT52, zum VT100 und zum VT200 und kann somit für den neuen 32-Bit-Rechner von Robotron K 1820 verwendet werden (dargestellt werden können 132 Zeichen pro Zeile).
(VEB Robotron-Elektronik Zella-Mehlis, Straße der Antifa 63–66, Zella-Mehlis, 6060; Tel. 6 10)



3



4

Die Pädagogische Hochschule „Karl Liebknecht" Potsdam informierte, wie Schüler der Unterstufe auf die Arbeit mit dem Computer vorbereitet werden. Das Bild 4 zeigt, wie den Kindern die wichtigsten Computerbegriffe erläutert werden, bevor sie sich das erste Mal an den Rechner setzen. Hinweis: Mit dem Begriff Menü ist hier das fertige Essen gemeint und nicht das in der Computerwelt übliche Bild für die Pogrammauswahl.

(Pädagogische Hochschule „Karl Liebknecht" Potsdam, Institut für Lehrerbildung „Rosa Luxemburg", Koll. Müller, Friedrich-Ebert-Straße 4, Potsdam, 1560; Tel. 32 90)

Text und Fotos: H. Hemke, H.-J. Hill

SYSTEMS 89

Vom 16. bis 20. Oktober 1989 fand in München zum 11. Mal die High-Tech-Messe SYSTEMS statt. Auf einer Ausstellungsfläche von 111 000 m^2 waren rund 1500 Aussteller aus 27 Ländern vertreten. Damit avanciert die SYSTEMS zur zweitgrößten Computermesse in der BRD nach der CeBit in Hannover. Etwa 200 000 Besucher zählte die SYSTEMS 89, was insbesondere darauf zurückzuführen ist, daß neben der Präsentation neuer Produkte ein interessantes Kongreßprogramm stattfindet, das vor allem EDV-Profis anlockt. Neben den Schwerpunktthemen Telekommunikation und Softwarestrategien veranstaltete die Gesellschaft für Informatik ihren 3. Internationalen GI-Kongreß unter dem Titel „Wissensbasierte Systeme".

Für Besucher mit Interese an bestimmten Technologien standen verschiedene Fachseminare bereit, so u. a. zu den Themen Datenbank- und Expertensysteme, Electronic Publishing, LAN, Mensch-Maschine-Interaktion, CASE und Bürokommunikation. Im SYSTEMS-Studio wurden offene Diskussionsforen abgehalten, auf denen Fachleute aktuelle Themen aus der Welt der Informatik erörterten. Im Vordergrund stand dabei die fortschreitende Computerkriminalität.

Allgemeine Trends

Die SYSTEMS 89 zeigte deutlich, daß sich auf dem DV-Markt zur Zeit ein Strukturwandel vollzieht, der durch die fünf folgenden Markttrends charakterisiert wird:

1. Der traditionelle DV-Markt zeigt zunehmend Sättigungserscheinungen. Jedes Unternehmen in der BRD mit mehr als 100 Mitarbeitern hat mindestens einen Computer installiert. Auch das PC-Potential wird 1992 zu mehr als 60 Prozent ausgeschöpft sein. Aus den enger werdenden Märkten resultieren zwangsläufig Überkapazitäten, die einen wachsenden Preiswettbewerb verursachen. Verstärkt wird dieser Trend durch das Bemühen, über standardisierte Systeme – Stichwort Unix – in die Märkte von Konkurrenten einzudringen.

2. Die Abhängigkeit der Informatikindustrie von der allgemeinen Konjunkturlage nimmt zu. Der Anteil des Ersatzgeschäfts und des Erweiterungsgeschäfts wächst. Das Investitionspotential für die Rationalisierung sich wiederholender Arbeitsverläufe (z. B. Rechnungswesen, Textverarbeitung usw.) ist weitgehend ausgeschöpft, so daß nur noch in geringem Umfang mit antizyklischen Investitionen zu rechnen ist, die Kostensenkungen herbeiführen sollen. Der Einsatz der Informatik dient heute vorrangig der Fitneßsteigerung im Wettbewerb. Und so zeigt sich, daß vom Konjunkturaufschwung erfaßte Wirtschaftszweige, wie der Maschinenbau oder die Finanzdienstleistungsbranchen, im Moment zu den besten Kunden der Informatikindustrie zählen.

3. Eine Verlagerung in andere, zum Teil neue Wachstumsmärkte ist erkennbar. Dazu gehören auf der einen Seite der weitgefächerte Markt für Dienstleistungen, auf der anderen Seite die jungen Märkte für Elektronikausrüstungen verschiedener Investitions- und Konsumgüter wie Werkzeugmaschinen, Kraftfahrzeuge, Waschmaschinen.

4. Am Dienstleistungsmarkt entsteht neues Potential für den Aufbau einer umfassenden informationstechnischen Infrastruktur (Software, Beratung, Systemintegration). Infrastrukturdienste auf der Netzebene einschließlich der sogenannten Mehrwertdienste (Value-added Network Services) werden zunehmend gefragt. Unabhängig davon entwickeln sich neue Märkte für Konsumelektronik, Fahrzeugelektronik, Medizinelektronik und Industrieelektronik.

5. Neben diesen Langzeittrends gibt es auch Sonderkonjunkturen und florierende Teilmärkte, wie z. B. die Märkte für Standardsoftware, Werkzeuge für die Softwareentwicklung (Stichwort CASE), UNIX-Systeme und Workstations sowie Lokale Netze und Telefax-Geräte.

Kaum noch Unterschiede zwischen PC und Workstation

Leistungsmäßig gibt es kaum noch Unterschiede zwischen den beiden Rechnergenerationen Personal Computer und Workstation. Die Definition der beiden Geräteklassen besagte lange Zeit: Personal Computer basieren auf einem 80x86-Prozessor und sind als Single-User-Maschinen im Einsatz. Workstations hingegen hatten meist einen Mikroprozessor der Marke 680x0 als CPU und waren als Multi-User-Systeme im Einsatz. Während bei den PCs das Betriebssystem MS-DOS im Vordergrund stand, zeichneten sich die Workstations durch das Betriebssystem Unix aus. Lange Zeit wurden die PCs nur im kommerziellen Bereich, die Workstations im technisch-wissenschaftlichen Markt eingesetzt. Heute verschwimmen die Grenzen: Die Hardware selbst wird immer unbedeutender.

Seit der Ankündigung des neuesten Intel-Prozessors 80486 steht fest, daß die Personal Computer in Leistungsbereiche verstoßen, die lange Zeit nur den Workstations vorbehalten waren. Computer mit dem 80486-Prozessor bieten Leistungen von 15 bis 25 Millionen Instruktionen pro Sekunde (kurz MIPS genannt). Die gängigsten Workstations liegen heute im Bereich von 14 MIPS. Zur SYSTEMS wurden die ersten Rechner mit dem neuesten Prozessor gezeigt, so von NCR, IBM und Rein Elektronik. Auch bei den traditionellen Workstations ist mit einem Leistungsschub in Richtung 20 MIPS zu rechnen. Während die Hardware kaum noch den Unterschied zwischen den PCs und den Workstations ausdrückt, spielt die Software eine immer bedeutendere Rolle. Auf den neuen PCs sind sowohl MS-DOS- als auch Unix-Programme lauffähig. Zwar bieten die Geräte unter MS-DOS, das für nicht so leistungsfähige Anwendungen und Prozessoren ausgelegt ist, die Möglichkeit, auch „alte, gewohnte" Software zu benutzen, die maximale Leistung des Systems wird dabei jedoch nicht ausgenutzt. Auf Dauer wird sich in diesem Bereich mit OS/2 ein anderes Betriebssystem für die Einzelplatzrechner durchsetzen. Auf der SYSTEMS wurden auch die ersten Programme gezeigt, die die neue Benutzeroberfläche Presentation Manager voll unterstützen.

Workstations hingegen laufen schon immer unter Unix. Durch die Standardisierungsbemühungen verschiedener Gremien gibt es sowohl im technischen/wissenschaftlichen als auch im kommerziellen Markt heute genügend Software unter diesem Betriebssystem.

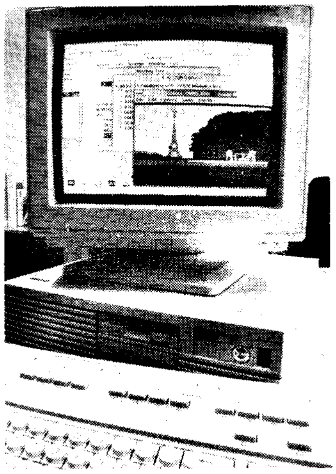
Wie die PCs sind auch die Workstations derzeit kurz vor einem Leistungssprung. Dieser wird durch die relativ neue Prozessortechnologie RISC (reduced instruction set computer) erzielt werden; gerade im technisch-wissenschaftlichen Sektor werden verstärkt RISC-Systeme angeboten. Das Betriebssystem Unix läuft ebenfalls auf diesen schnellen Systemen. Während der RISC-Markt bei Workstations, rein von den Umsätzen her betrachtet, im vergangenen Jahr gerade 10 Prozent ausmachte, zeigen die Ankündigungen der letzten Monate, daß das Angebot an RISC-Rechnern immer größer wird. Für RISC-Systeme gibt es bereits ein breitgefächertes Softwareangebot. Das Betriebssystem Unix wird verstärkt auch im kommerziellen Bürobereich Anwendung finden.

Mit dem 1987 von IBM vorgestellten Mikrokanal und den darauf basierenden PS/2-PC-Systemen gab es erstmals eine zur ISA (Industrie Standard Architektur) konkurrierende PC-Architektur. Auf der SYSTEMS 89 wurde von der Fa. Hewlett Packard der weltweit erste Rechner mit EISA-Architektur vorgestellt. EISA (Erweiterter Industrie Standard Architektur) als Konkurrenz gegenüber dem IBM-Mikrokanal hat gegenüber ISA folgende Vorteile:

– volle Rückwärtskompatibilität mit ISA-Systemkomponenten
– voller 32-Bit-Adreß- und 32-Bit-Datenbus
– automatische Konfiguration für Erweiterungssteckkarten
– verbesserte DMA-Funktionen (Direct Memory Access) mit Durchsatzraten bis zu 33 MByte/s
– Unterstützung intelligenter Peripheriegeräte durch die Möglichkeit, einen parallel zum Hauptprozessor arbeitenden Bus-Master mit eigenem Prozessor einzusetzen
– zusätzliche Funktionen, die die Entwicklung komplexer, intelligenter Peripheriegeräte ermöglichen.

So besitzt der von HP präsentierte Vectra 486 beispielsweise folgende technische Daten:

– Prozessor Intel 80486 mit 25 MHz Taktrate



Siemens erweiterte zur SYSTEMS seine Produktpalette um zwei weitere 80386-Modelle. Der hier gezeigte PCD-3M ist insbesondere für Anwendungen mit grafischen Benutzeroberflächen – beispielsweise MS-Windows – gedacht. Als Einstieg in die Welt der 386er wird zu einem geringeren Preis der PCD-3M sx angeboten.

Der zur CeBIT 89 angekündigte PC 50-II mit dem Prozessor 80386 SX wurde von Commodore erstmals zur SYSTEMS vorgeführt. Er verkörpert nach den Worten von Peter Kaiser, neuer Geschäftsführer der Commodore Büromaschinen GmbH, die konsequente Produktorientierung von Commodore. Der PC 50-II wird in drei Versionen angeboten und kann je nach Ausstattung Aufgaben als Workstation, Arbeitsplatzrechner oder Fileserver übernehmen.

– 8 EISA-Steckplätze wahlweise für 8-, 16- oder 32-Bit-Zusatzplatinen
– 1 parallele, 2 serielle Schnittstellen
– Tastatur- und Mausanschluß
– 2 MByte RAM, aufrüstbar bis 64 MByte
– Shadow-RAM für BIOS-Code
– 64 KByte ROM
– Festplattenkapazität ausbaufähig bis 670 MByte
– Diskettenlaufwerk 5,25"; 1,2 MByte
– 120 MByte Tape-Back-up
– Monitore von MDA bis VGA sowie HP Intelligent Graphics.

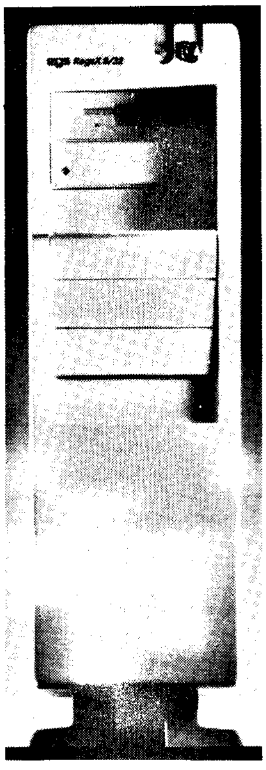
Als Betriebssysteme sind einsetzbar: MS-DOS 4,01, Windows/286 oder / 386, OS/2, SCO UNIX /386 V 3.2 und VP/ix 1.2.
Dieser Hochleistungsrechner eignet sich vor allem für anspruchsvolle CAD-Anwendungen mit hochauflösender Grafik und als Fileserver.
Zur Zeit werden EISA-Rechner von acht großen Computerherstellern unterstützt und weiterentwickelt: AST Research INC., COMPAQ Computer Corporation, EPSON America Inc., HEWLETT-PACKARD Company, NEC Information Systems Inc., OLIVETTI, TANDY, WYSE Technology und ZENITH Data Systems Corporation.
Nach den auf der SYSTEMS gemachten Vorankündigungen stehen z. B. von OLIVETTI und COMPAQ entwickelte EISA-Rechner kurz vor ihrer Präsentation. Damit ist eine neue Entwicklungsstufe im PC-Bereich erreicht.
Eine Weltpremiere gab es auch von Toshiba. Der Laptop-Marktführer stellte den ersten serienreifen 14-Zoll-Farb-LCD-Schirm vor. Dieser ist in zwei Modi betreibbar: 720 × 550 Pixels bei 16 Farben und 1440 × 1100 Pixels im Schwarz/Weiß-Modus.
Parallel dazu zeigte SHARP ebenfalls einen Laptop mit 14-Zoll-Farb-LCD, den PC-8041. Der auf einem 80386 basierende tragbare Computer arbeitet mit einer Taktfrequenz von 20 MHz und bietet volle VGA-Emulation. Das Gewicht beträgt weniger als 10 kg.

Neue Speichertechnologien ändern die DV-Welt

Ob bei Großrechnern, Workstations oder Personal Computern: Immer mehr DV-Hersteller bieten optische Platten zur Datenhaltung an. Mit ihrer riesigen Speicherkapazität auf kleinstem Raum verändern die „Optical disks" nachhaltig die gesamte DV-Landschaft. Aller Anfangseuphorie zum Trotz taten sich die optischen Speichermedien bisher recht schwer im DV-Markt. Das wird sich in den nächsten Jahren jedoch grundlegend ändern. Zum einen ist die Technik nun ausgereift, zum anderen einigten sich die Hersteller auf gemeinsame Standards. Den Durchbruch der optischen Speichermedien auf breiter Front erwarten Marktbeobachter daher für die nächsten Jahre. So soll sich der Verkauf der CD-ROM-Platten nach einer Prognose von Link Ressources in Europa allein im Zeitraum 1988 bis 1992 von 25 000 auf 280 000 Stück mehr als verzehnfachen. Die CD-ROM entspricht in ihren Abmaßen der Audio-CD (5,25-Zoll-Laufwerk, 120 mm Plattendurchmesser) und ist wie diese nur lesbar und nicht wiederbeschreibbar (Compact Disk – Read Only Memory). Als Datenträger bietet sie immense Vorteile: Eine einzelne Platte faßt bis zu 650 MByte – eine Floppy dagegen nur maximal 1,44 MByte. Sie ist unempfindlich gegen Staub und Headcrashs (also stoßunempfindlich) und speichert neben den üblichen DV-Daten auch Zeichnungen, Bilder, Musik und Sprache. Die glänzenden Platten werden heute überwiegend als Informationsträger für Nachschlagewerke und umfangreiche Kataloge (vom Telefonbuch bis hin zu Patentschriften und Medikamentenverzeichnissen) eingesetzt. Über 800 CD-ROM-Anwendungen wurden 1988 bereits weltweit gezählt. CD-ROM-Laufwerke für Personal Computer kosten derzeit noch zwischen 1 000 und 2 000 DM.
Bereits im Einsatz ist heute ebenfalls die mit Anwenderdaten einmal beschreibbare und immer wieder lesbare optische Platte, die sogenannte WORM (Write Once Read Many). Diese Platten eignen sich vor allem zur Datensicherung und Archivierung von großen Datenmengen. Gegenüber den bisher benutzten Medien wie Band und Kassette besitzen sie eine vielfach höhere Speicherfähigkeit. Verwaltet in „Juke Boxes" lassen sich Kapazitäten bis zu 300 GByte und mehr erreichen. Das entspricht etwa sechs Millionen A4-Seiten. WORM-Platten lösen in vielen Fällen auch den Mikrofilm als Archivmedium ab.
Weitgehend gelöst sind heute auch die technischen Probleme bei mehrfach beschreibbaren optischen Disks. Auf der SYSTEMS 89 zeigten verschiedene Hersteller entsprechende Produkte im 5,25-Zoll-Format.

PC-Netze bilden das Rückgrat der Unternehmenskommunikation

Wie die SYSTEMS zeigte, hat sich in der Praxis ein dreistufiges Konzept der Unternehmenskommunikation durchgesetzt. Dabei arbeiten auf der untersten Ebene Personal Computer, die untereinander und mit den beiden



RegsX 8/32 ist ein Multiprozessorsystem von EGS für Unix-Anwendungen. Es können bis zu 8 masterslavefähige 32-Bit-Prozessorkarten auf Basis des Motorola 68020/ 68030 mit einer Gesamtleistung von 40 MIPS installiert werden. Durch die Integration eines LAN-Schnittstellen-Konzentrators für 8 Terminals eignet sich das RegsX 8/32 besonders als Fileserver in Unix-Netzen.

„übergeordneten" Stufen – den Minicomputern auf der Abteilungsebene und den Großrechnern – vernetzt sein können. Für lokale Netze haben sich Standards wie die Bus-Netz-Topologien von Token Bus und Ethernet sowie die Ring-Topologie des Token Ring-Standards durchgesetzt, wobei Token-Verfahren gerade bei hohen Anforderungen und hohen Datenraten vorgezogen werden. Neben dem OSI-Referenzmodell entsprechenden Normen haben sich dabei zum Teil auch De-facto-Standards wie etwa TCP/IP als Kommunikationsschnittstelle durchgesetzt. Sollen mehrere Netze miteinander verbunden werden, so kommen sogenannte Router, Bridges und/oder Gateways zum Einsatz, die – falls erforderlich – die in der Regel darüber liegenden (OSI-)Kommunikationsprotokolle umsetzen können. Die Funktionalität dieser Kopplungskomponenten ist mit bestimmten Netzverwaltungsaufgaben verbunden. Server sowie Programme zur Netzüberwachung und -steuerung sowie Anwendungssoftware für Multiuser- und Multitasking-Umgebungen machen aus den einzelnen Elementen eine integrierte PC-Netz-Anwendung.
Der Informationsbedarf der Anwender in Sachen „lokale Netzwerke" steigt mit dem Grad der Durchdringung aller Unternehmensbereiche mit Computertechnik und deren Komplexität. Die SYSTEMS 89 gab einen Überblick über das gesamte Spektrum der betrieblichen Kommunikation. Standards und Normen der Kommunikationsstrukturen sowie alle Produkte, die diesen Normen entsprechen, wurden ebenso gezeigt wie beispielhafte Lösungen für eine große Anwendungspalette. Verkabelungsstrategien können die Kommunikationsinfrastruktur häufig nachhaltig beeinflussen und waren daher neben einer umfassenden Darstellung hybrider Netze für die Sprach- und Datenkommunikation Gegenstand eines Fachseminars im Kongreßteil der SYSTEMS.

Einheitliche Benutzeroberflächen setzen sich durch

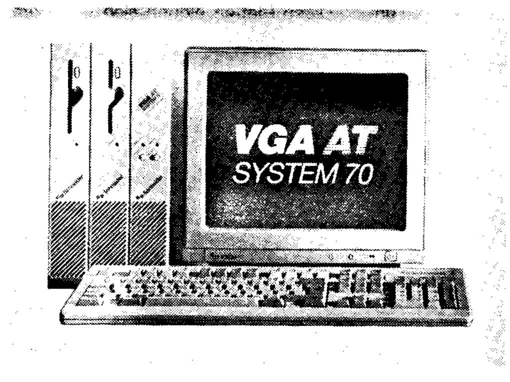
Wie die SYSTEMS zeigte, sind grafische Benutzeroberflächen stark im Kommen. Die derzeit populärste Oberfläche stammt von dem Massachusetts Institut of Technology, MIT, das X.11 bzw. X.Windows zusammen mit einer großen Zahl von Computerherstellern entwickelte. Der Vorteil einer einheitlichen Benutzeroberfläche ist für den Anwender sehr erheblich. Wer viele unterschiedliche Programme auf seinem Rechner nutzt, kennt das Problem: Jede Software verlangt andere Eingaben, und die Funktionstasten haben andere Bedeutungen, Hilfestellungen werden mit verschiedenen Befehlen angefordert. Mit einer Standardoberfläche erscheinen alle Programme einheitlich. Heute wird X/Windows von einer großen Anzahl von Softwareanbietern unterstützt. Selbst der Macintosh kann mit X/Windows arbeiten. X/ Windows läuft derzeit auf PCs, Workstations und auch auf Minicomputern. Der Umstieg von einer Hardware auf eine andere Plattform bedeutet für den Anwender, daß er eine ihm bekannte Oberfläche auf allen Systemen vorfindet. Die Einarbeitungszeit in unterschiedlichen Programme wird dadurch erheblich verkürzt – die Akzeptanz für neue Produkte erhöht. Die berühmte Fenstertechnik kostet allerdings eine Menge Hauptspeicher. Und der ist derzeit recht teuer. Auch ist die Anzahl der heute verfügbaren Programme unter X/Windows noch recht mager. Zwei Varianten von X/Windows haben auf der SYSTEMS 89 in München Furore gemacht: Open Look und OSF/Motif. Bei Open Look handelt es sich um

eine Benutzeroberfläche, die aus dem Hause AT & T stammt und derzeit auf Unix-Rechnern läuft. Auch OSF/Motif läuft unter Unix; im Januar wurde dieses leistungsstarke User-Interface der Öffentlichkeit erstmalig vorgestellt, und auf der SYSTEMS 89 konnten die ersten Software-Pakete unter OSF/Motif in Augenschein genommen werden. Während in den Unix-Gremien, die sich um eine Standardisierung dieses Betriebssystems bemühen, derzeit heftigst über Open Look und Motif diskutiert wird, scheinen die Softwarehäuser zur Zeit Motif zu favorisieren. Letztendlich wird sich nur das Unser-Interface durchsetzen, auf das die Softwarehäuser und Hardwarehersteller setzen.

Computer Aided Software Engineering – ein Werkzeug zum Abbau des „Anwendungsstaus"

Der Begriff „Anwendungsstau" ist mindestens ebenso verbreitet wie der Begriff CASE – nur daß dort, wo CASE (computer aided software engineering) professionell praktiziert wird, der Anwendungsstau erheblich kleiner als anderwärts ist, ja manchmal überhaupt nicht in Erscheinung tritt. Die Erkenntnis, daß CASE trotzdem kein simples Allheilmittel darstellt, sondern unbedingt problem- und situationsgerecht ausgewählt werden muß, setzt sich in der Fachwelt inzwischen immer stärker durch. Einen aktuellen Überblick über die neuesten Entwicklungen im CASE-Sektor konnte man auf der SYSTEMS erhalten. So konnte sich der Fachbesucher beispielsweise darüber informieren, ob und wie das jeweilige Werkzeug ihn bereits vor der Erstellung des Quellencodes unterstützt, wie viele Phasen des Software-Entwicklungsprozesses es abdeckt, und ob es in einer deutschsprachigen Version angeboten wird. Man konnte Front-End-, Back-End- und projektbegleitende CASE-Werkzeuge und auch solche Tools kennenlernen, die zur Sanierung von Alt-Software herangezogen werden können. Es wurden Systeme gezeigt, die als Data-Dictionary fungieren und das Problem der Software-Dokumentation elegant und einfach lösen. Die methodischen Hintergründe wurden erläutert, auf denen die jeweiligen Tools aufsetzen. Denn die Kenntnis der von den einzelnen Werkzeugen unterstützten Methoden ist eine wichtige Voraussetzung dafür, daß der Anwender die Produktivität seiner Software-Entwicklung gezielt steigern kann.

Die SYSTEMS 89 bot dem Anwender aber auch die Möglichkeit zu ermitteln, welche externen Beratungsunternehmen ihm im schwer überschaubaren CASE-Umfeld welche Hilfestellung leisten können. Derartige Beratungsleistungen umfassen in der Regel eine Situationsanalyse, der sich eine Einführungsstrategie und ein Trainingskonzept anschließen. Sodann konzentrieren Berater und Anwender sich auf das Vorgehensmodell und das Methodensystem, auf die Auswahl und den Einsatz der Tools sowie auf die technische CASE-Implementierung. Meist übernimmt der Berater auch Aufgaben der Projektbegleitung und der Mitarbeiterschulung. Auf der SYSTEMS konnte sich der DV-Anwender davon überzeugen, daß CASE bereits zu einem Instrumentarium für die Software-Entwicklung einschließlich Rapid Prototyping herangereift ist, das heißt, daß die Durchgängigkeit der Entwicklungsarbeiten und ihrer Resultate von der Planung eines Softwareprojekts bis hin zur Inbetriebnahme und Wartung mittlerweile erreicht wurde.



Für das wachstumsträchtige Marktsegment der gewerblichen Anwender bei Klein- und mittelständischen Unternehmen bietet Schneider neben dem Tower AT jetzt die neuen Produkte VGA AT System 40 (42-MByte-Festplatte) und VGA AT System 70 (68-MByte-Festplatte) an. Gemeinsam ist ihnen der VGA-Videoadapter, der kompatibel zu den Standards VGA, EGA, CGA, MDA und Hercules ist und eine Grafikauflösung von bis zu 1024 × 768 Punkten erreicht.

Der bekannte Tastatur- und Schalterhersteller Cherry zeigte eine IBM-MF-kompatible Tastatur, die als Besonderheit mit einem Magnetkarten-Durchzugsleser ausgestattet wurde. Für den Betrieb ist keine zusätzliche Software erforderlich; die Leseeinheit arbeitet über die Tastaturschnittstelle und blockiert somit keine weitere Schnittstelle.

Software und Service gewinnen gegenüber der Hardware immer stärker an Gewicht

Für Unternehmen und Institutionen spielt die Verfügbarkeit geeigneter Software eine immer größere Rolle. Dies gilt ebenso für Einzelpersonen, die beruflich (auch etwa zu Ausbildungszwecken) auf qualitativ hochwertige Software angewiesen sind. Die SYSTEMS 89 informierte umfassend über die Möglichkeiten des Erwerbs oder der Entwicklung solcher Software und vermittelte in zahlreichen Symposien, Seminaren und Kongressen Einblick in die Leistungsbreite wie auch in die Problematik moderner Software.

Aus der Fülle der angebotenen Software, seien es Neuentwicklungen oder Erweiterungen bewährter Software, ist es unmöglich, an dieser Stelle auf einzelne Programme näher einzugehen. Besondere Erwähnung verdient allerdings der Markt der Unix- und Unix-nahen Systeme. Experten prognostizieren für ihn in den nächsten Jahren gewaltige Zuwachsraten. Aktuelle Neuentwicklungen bestätigen die Dynamik dieses Vorgangs: So nimmt die Zahl der Computersysteme, auf denen Datenbankmanagementsysteme unter Unix lauffähig sind, sprunghaft zu. Immer mehr Datenverbindungen zwischen Unix und anderen Softwaresystemen (MS-DOS, VMS, Datex-P usw.) werden geschaffen. Die Vielfalt der Standardanwendungen unter Unix wächst ständig – bis hin zu Realtime-Applikationen, die unter Unix entwickelt werden.

DV-Normen: Impulse für die Datenverarbeitung

Die SYSTEMS 89 gab einen aktuellen Überblick über den Stand der Normierung in verschiedenen Teilbereichen der Informations- und Kommunikationstechnik und das Angebot an entsprechenden Produkten. Die DV-Landschaft wird übersichtlicher. Nach Jahrzehnten des Produktwirrwarrs und der Inkompatibilität stehen Standards beim Anwender ganz oben auf der Wunschliste. Vorreiter dieser Entwicklung hin zu einer einheitlichen elektronischen Sprache sind die Behörden. So wurde in der EG im Februar 1988 beschlossen, bei öffentlichen Aufträgen auf dem Gebiet der Informationstechnik nur noch Produkte einzusetzen, die auf internationalen oder europäischen Normen oder Vornormen basieren. Bereits heute kann der Anwender auf eine Vielzahl von Normen mit entsprechenden Produkten zurückgreifen. Auf der Betriebssystemebene etabliert sich zusehends Unix als der Standard, wobei sich die X/Open-Vereinigung um eine Vereinheitlichung der verschiedenen Ansätze bemüht. Es gibt heute kaum mehr einen Computerhersteller, der keinen Unix-Rechner im Programm führt. Auf der Kommunikationsseite ist mit OSI (Open Systems Interconnection) ein offenes Netzwerk bereits als Norm verabschiedet, das eine Kommunikation zwischen Rechnern der verschiedensten Hardwarehersteller erlaubt. Bereits im Dienst sind OSI-Anwendungen wie X.400 für die elektronische Post. Kurz vor der allgemeinen Einführung stehen Produkte für den standardisierten Austausch von Geschäftsdaten mit EDIFACT (Electronic Data Interchange for Administration, Commerce and Transport) und dem Austausch von Bürodokumenten nach ODA (Office Document Architecture). Interessant für DV-Anwendungen ist auch das künftige integrierte digitale Netz der Deutschen Bundespost (ISDN).

Standards bieten für die Kunden eine Reihe von Vorteilen: Reduzierung der Hersteller-Abhängigkeit, bessere Vergleichbarkeit der Angebote, Senkung der Preise mit zunehmendem Wettbewerb, höhere Investitionssicherheit; nicht zu vergessen auch die Senkung der Wartungs- und Schulungskosten.

Für die Hersteller bieten die Standards die Möglichkeit, neue Kundenkreise und Anwendungen zu erschließen. In der Regel ist der Umstieg auf Standards mit Neu-Investitionen verbunden, wodurch der DV-Markt neue Impulse erhält. *H. Benicke*

Erste europäische Amiga-Computermesse

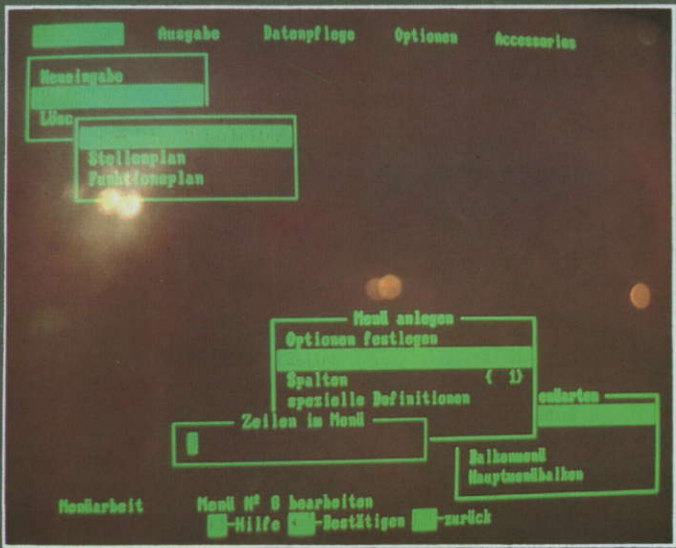
Für die Amiga-Freunde in Europa gibt es jetzt eine eigene Computermesse. Nach dreitägiger Dauer ging Mitte November in Köln die erste europäische Amiga-Messe zu Ende; die nächste, Amiga '90, wurde bereits für den 8. bis 11. November 1990 angekündigt, mit dann voraussichtlich rund einhundert Ausstellern und etwa verdoppelter Ausstellungsfläche. In diesem Jahr zählte der Veranstalter, die amerikanische Organisation AmiEXPO, über 35000 zahlende Besucher. 77 Aussteller aus 6 Ländern, darunter 17 USA-Aussteller, nutzten die Europa-Premiere, um Entwickler, Anwender, Groß- und Einzelhändler zu treffen.

Commodore als Hersteller des Amiga und Schirmherr der Messe sieht sich damit in der Auffassung bestätigt, daß aus dem Kreis der Amiga-Fans mehr und mehr Problemlösungen auf Basis dieser Produktfamilie entwickelt werden. Parallel dazu will Commodore mit der Einführung der 2500er-Reihe die Weichen für eine weitere Expansion im professionellen Markt stellen. *MP*

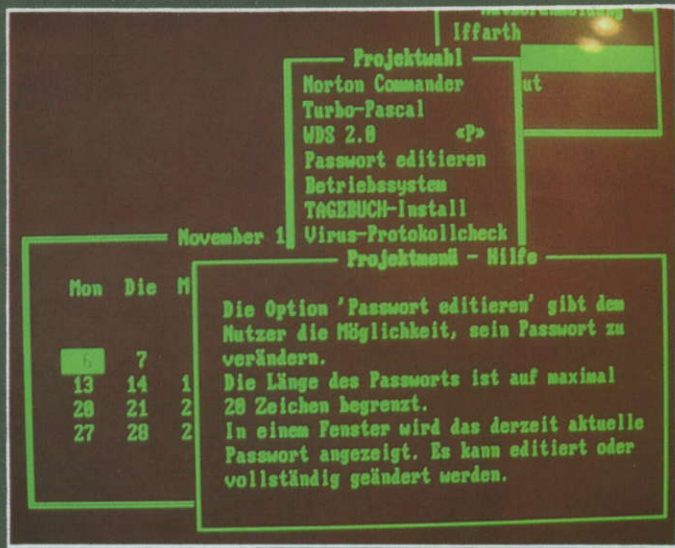
TERMINE

Kommunikationskongreß KoKon '90

WER? KommIntern – die KommunikationsInternationale
WANN? 24. und 25. Februar 1990
WO? Berliner Haus der Jungen Talente
WAS? Unter dem Motto der Bürgerdiplomatie geht es um Themen wie kreative Techniknutzung, Datenschutz als Menschenschutz, um Informationsökologie, die Macht der Medien im globalen Dorf, das Verfassungslos, den Freedom of Information Act, das Recht zum Kopieren, um Datenreisefreiheit, Viren und andere Getüme.
WIE? Teilnahmemeldungen an HdJT-Computerclub, Stefan Seeboldt, Klosterstraße 68 bis 70, Berlin, 1020; Tel. 2103265 *Seeboldt*

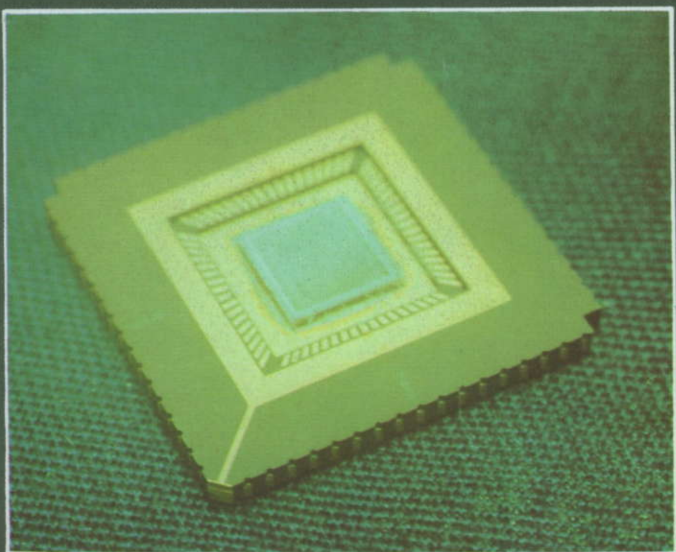


① Wer nicht Zeit noch Lust hat, sich tagelang mit dem Entwurf von Menüstrukturen und deren Farbgestaltung zu befassen, kann das leistungsfähige Window-Development-System nutzen.



② Mit einer komfortablen Nutzerführung wurde das I&T Maschinentagebuch für einen derzeit noch typischen DDR-Betrieb entwickelt. Neben einem gedruckten Bericht über die kalendertägliche Auslastung des PCs für Ihren Chef übernimmt das System auch den passiven Virenschutz.

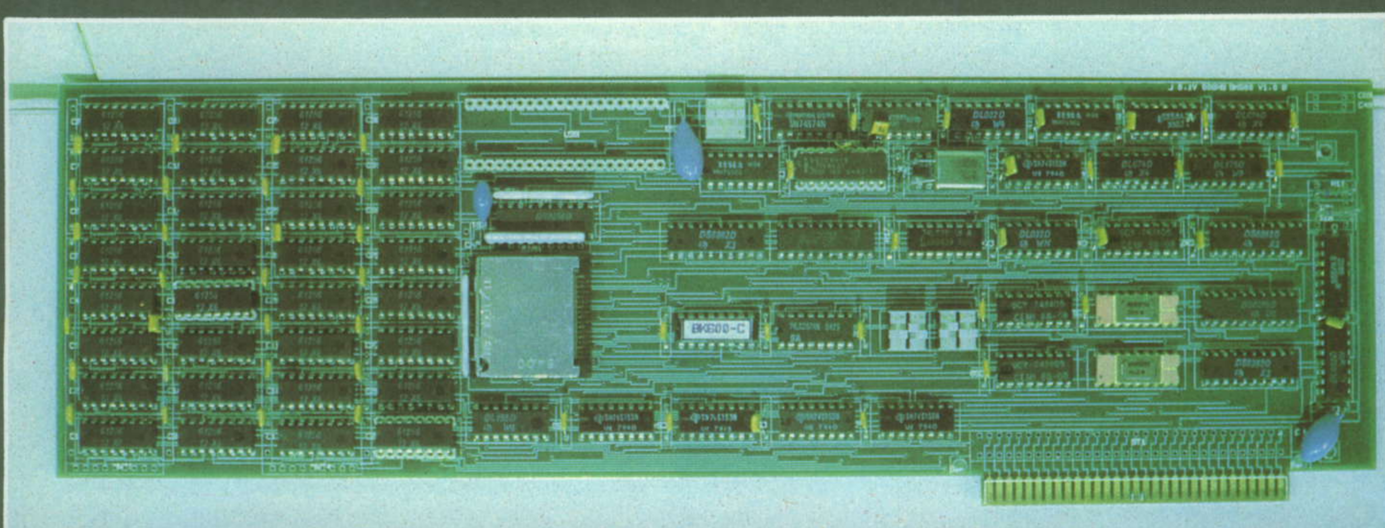
Bilder von der 32. ZMMM



③ Der U 80617 ist ein Busanpassungsschaltkreis für den Ethernetcontroller des 32-Bit-Rechners K 1820 von Robotron.

④ Mit der Großfeldtastatur für Körperbehinderte wird das Anwendungsspektrum von Kleincomputern vergrößert. Die Buchstabenanordnung ermöglicht eine Senkung der Zahl der Fingerwege auf 55 Prozent und der Schreibzeit auf 60 Prozent.

⑤ Mit der Turbo-Karte BK 600 wird der EC 1834 bei einer Taktfrequenz von 8 MHz auf das 3- bis 4fache und mit dem Koprozessor U 80613 sogar auf das 5- bis 8fache beschleunigt.

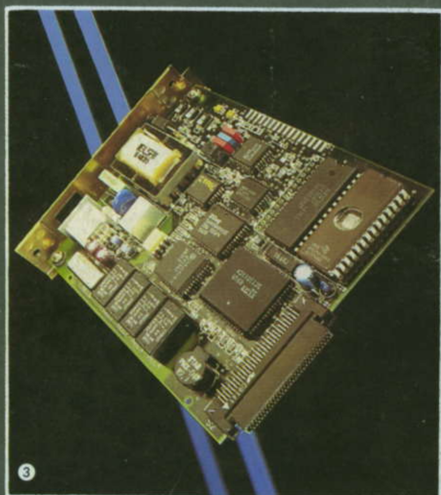


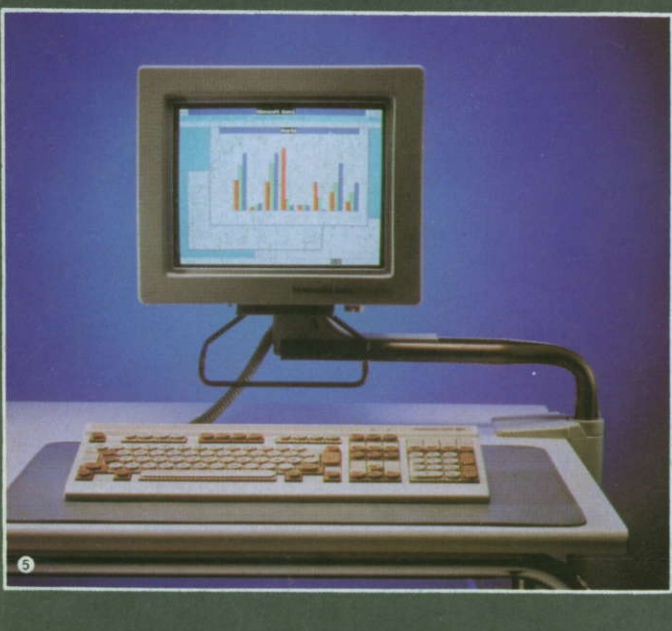
ISSN 0232-2892
Mikroprozessortechnik, Berlin 4 (1990) 2
S. 33–64

Neuheiten auf der SYSTEMS 89

Lesen Sie dazu auch unseren Bericht in diesem Heft.

① Mit dem Flaggschiff ihrer 32-Bit-PCs sieht sich MITAC „an der Spitze der schnellsten Rechner der Welt". Der mit 33 MHz getaktete und ohne Wartezyklen arbeitende Prozessor Intel 80386 verleiht dem MPC 4000G eine Leistung von 8,3 MIPS (Millionen Instruktionen je Sekunde). Damit ist der PC für anspruchsvolle CAD-Aufgaben oder als LAN-Server besonders geeignet.





② Das Towermodell AT 80386-33 von PEACOCK ist ein weiterer Vertreter der mit 33 MHz getakteten 32-Bitter. Eine voll aufgerüstete Variante mit 8 MByte RAM, 64 KByte Cache und 380 MByte Festplattenkapazität eignet sich besonders für den Einsatz als LAN-Server.

③ Toshiba stellte zur SYSTEMS das Einschubmodem Microlink 2400 für seine Laptop-Modelle ab dem T 1200 vor. Es eignet sich für Datenfernübertragung nach CCITT-Norm, einschließlich Telefax. Das Microlink 2400 arbeitet nach dem erweiterten „Hayes AT-Standard", der sich als de-facto-Standard bei Wählmodems durchgesetzt hat.

④ Eine besondere LAN-Workstation kommt von ADCOMP. Die Netcom 1600 im Miniformat integriert CPU, Disk-I/O, serielle und parallele Schnittstellen, Grafik-Controller und LAN-Interface-Karte auf einem einzigen Board im CMOS-Design. Die Netcom 1600 kann sowohl in Arcnet- als auch in Ethernet-Netzen zum Einsatz kommen.

⑤ Eine neue Generation von ergonomischen Farbmonitoren stellte TANDBERG DATA mit der 5000er Serie vor. Der 5320 ist ein VGA-kompatibler Monitor mit einer Auflösung von 800 × 600 Pixeln. Er ist anschließbar an alle IBM PC/AT/PS/2-kompatiblen PCs.

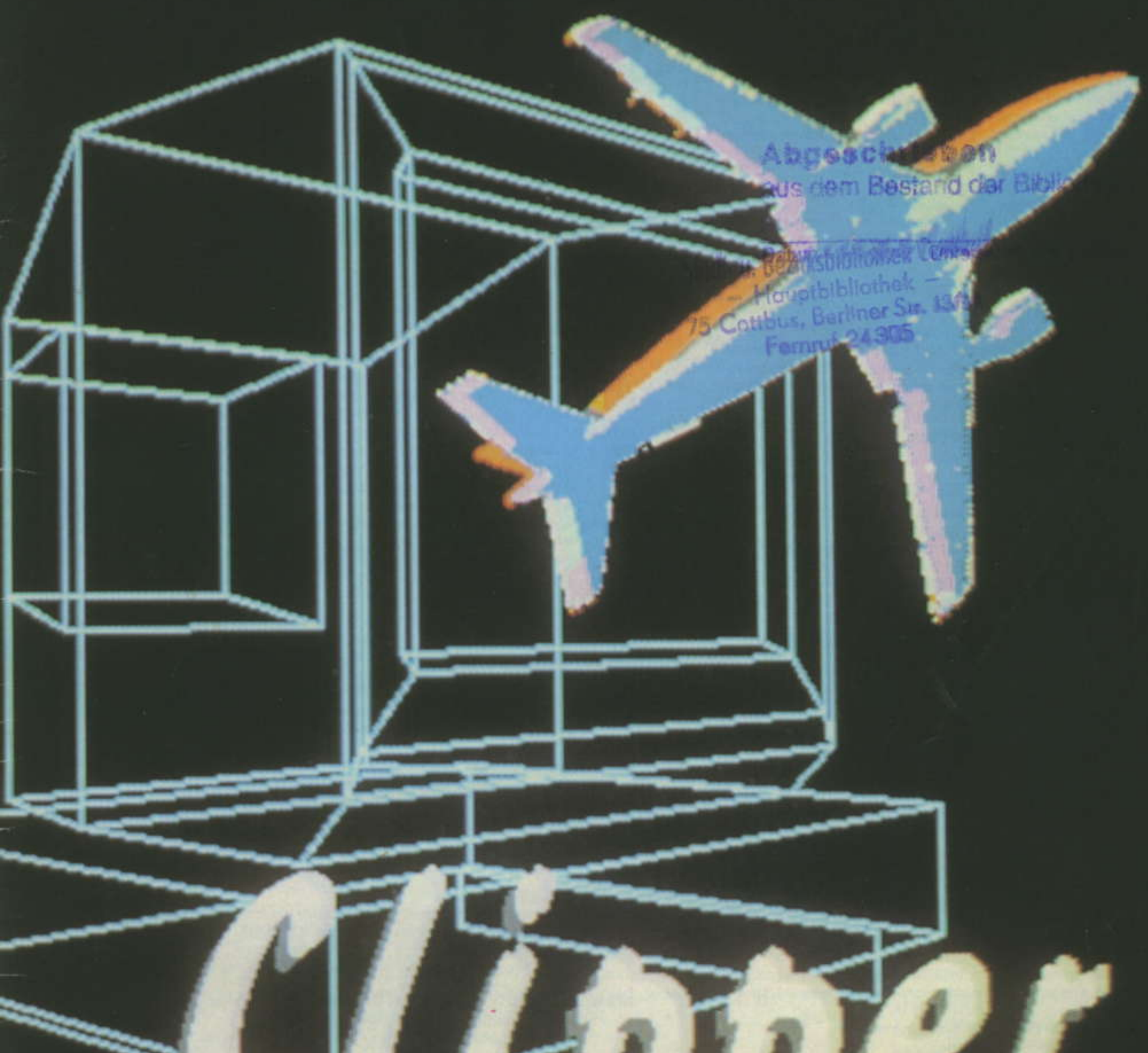
ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

MP

Heft 3 · 1990

Mikroprozessortechnik

VEB Verlag Technik Berlin
ISSN 0232-2892



erstmals mit Beiträgen aus c't

LOCATE [<gb>] **FOR** | **WHILE** <bedingung> Positionieren des Datensatzzeigers auf den ersten Datensatz des Datenbankfiles, der die Bedingung erfüllt.
LOOP Bewirkt innerhalb einer DO-WHILE-Anweisung die Übergabe der Programmsteuerung an den Anfang dieser DO-WHILE-Anweisung.

MENUE TO <speichervar> Abschließende Eingabeanweisung für private Menüs, die zuvor durch @ ... PROMPT aufgebaut wurden.

NOTE | * <text> Kommentarzeile mit Makroersetzung.

PACK Physisches Löschen aller als gestrichen markierten Datensätze.
PARAMETERS <par_liste> Spezifizieren der Parameter einer Prozedur, einer Funktion oder eines Kommandofiles.
PRIVATE <var_liste> Beschränken der Verfügbarkeit von Speichervariablen oder Feldern.
PROCEDURE <proz_name>
[**PARAMETERS** <par_liste>]
<kommandofolge>
[**RETURN**]
Definition einer Prozedur.
PROMPT Siehe @ ... PROMPT.
PUBLIC <var_liste> Speichervariable, einschließlich Felder, werden global sichtbar.

QUIT | **CANCEL** Abbruch der Programmausführung, Schließen aller eröffneten Files und Rückkehr zum rufenden Programm.

READ [**SAVE**] Aktivieren des GET-Teils aller seit dem letzten READ bzw. dem Programmstart abgearbeiteten @ ... GET-Kommandos und Setzen des Kursors auf den Eingabebereich des ersten GET-Kommandos.
RECALL [<gb>] [**FOR** | **WHILE** <bedingung>] Streichen der Löschmarkierung von Datensätzen.
REINDEX Aktualisieren aller aktiven Indexfiles.
RELEASE <var_liste>
RELEASE ALL [**LIKE** | **EXCEPT** <gruppenbezeichnung>]
Streichen von Speichervariablen.
RENAME <alter_filename> | (<c_ausdr_1>) **TO**
<neuer_filename> | (<c_ausdr_2>)
Umbenennen eines Files.
REPLACE [<gb>] <feld_1> **WITH** <ausdr_1> {,<feld_i> **WITH** <ausdr_i>...}
[**FOR** | **WHILE** <bedingung>] Ersetzen von Feldinhalten.
REPORT FROM <filename> | (<c_ausdr_1>) [<gb>] [**FOR** | **WHILE** <bedingung>]
[**PLAIN**] [**NOEJECT**] [**SUMMARY**] [**HEADING**] <c_ausdr_2> [**TO PRINT**]
[**TO FILE** <filename> | (<c_ausdr_3>)]
Ausgabe eines Berichts.
RESTORE FROM <filename> | (<c_ausdr>) [**ADDITIVE**]
Laden von Speichervariablen aus einer Datei (.mem).
RESTORE SCREEN [**FROM** <speichervar>] Wiederaufbau eines Bildschirminhalts aus einer zuvor mit SAVE SCREEN geretteten Speichervariablen.
RETURN [<ausdr>] Rückgabe der Programmsteuerung an das aufrufende Kommandofile (Prozedur). Bei Funktionen Rückgabe des berechneten Wertes.
RUN | ! <DOS_kommando>| (<c_ausdr>) Aufruf eines DOS-Kommandos, oder Start eines weiteren Programms.

SAVE SCREEN [**TO** <speichervar>] Speichern des Bildschirminhalts.
SAVE TO <filename> | (<c_ausdr>) [**ALL** [**LIKE** | **EXCEPT** <gruppenbezeichnung>]]
Speichern aller ausgewählten Speichervariablen in ein File (.mem).
SAY Siehe @ ... SAY
SEEK <ausdr> Positionieren des Datensatzzeigers auf den ersten Datensatz, dessen Index gleich dem Wert des Ausdrucks ist.
SELECT <arbeitsbereich> | <alias> | (<c_ausdr>) Auswahl eines Arbeitsbereichs.
SET ALTERNATE TO [<filename> | (<c_ausdr>)] Definition eines Alternate-Files.
SET ALTERNATE ON | **OFF** (<l_ausdr>) An- und Abschalten der Ausgabe in das Alternate-File.
SET BELL ON | **OFF** | (<l_ausdr>) An- und Abschalten eines Signals, welches bei der Eingabe des letzten Zeichens eines Feldes ertönt.
SET CENTURY ON | **OFF** | (<l_ausdr>) Anzeige des Jahrhunderts, bei Datumsangaben.
SET COLOR | **COLOUR TO** {<standard>, [<direkt>],
[<rahmen>, [<hintergrund>] [,<nicht_gewählt>]]]
SET COLOR | **COLOUR TO** [<c_ausdr>) Auswahl der Anzeigearten und Farben auf dem Bildschirm.
Anmerkung: Beim Einsatz von alphanumerischen Bildschirmen sind folgende Einstellungen möglich: weiß (w), schwarz (n), unterstrichen (u), unsichtbar (x), intensiv (i), blinkend (*), grafikfähige Bildschirme können darüber hinaus folgende Farben bzw. Graustufen darstellen: blau (b), braun (gr), cyan (bg), gelb (gr+), grau (n+) und grün (g).
SET CONFIRM ON | **OFF** | (<l_ausdr>) Kursorsprung zum nächsten Eingabefeld.
SET CONSOLE ON | **OFF** | (<l_ausdr>) Ausgabe auf dem Bildschirm.
SET CURSOR ON | **OFF** | (<l_ausdr>) Bildschirmkursor ein-/ausschalten.
SET DATE AMERICAN | **ANSI** | **BRITISH** | **FRENCH** | **GERMAN** | **ITALIAN**
Festlegen der Datumsform.
SET DECIMALS TO <n_ausdr> Anzahl der Dezimalstellen für zu rundende Werte.
SET DEFAULT TO <gerät>[:<pfad>] | (<c_ausdr>) Standardgerät und Verzeichnis für den Zugriff auf Datenfiles.
SET DELETED ON | **OFF** | (<l_ausdr> Bearbeiten der als gelöscht markierten Datensätze.
SET DELIMITERS TO [<c_ausdr> | **DEFAULT**] Definieren einer Zeichenkette, die als Begrenzer für Eingabebereiche benutzt wird (Standard ":").
SET DELIMITERS ON | **OFF** | (<l_ausdr>) ON schaltet die Anzeige der durch SET DELIMITER TO definierten Zeichenkette auf dem Bildschirm ein.
SET DEVICE TO SCREEN | **PRINT** Die Ausgaben der @-Kommandos werden auf dem Bildschirm oder dem Drucker ausgegeben.
SET ESCAPE ON | **OFF** | (<l_ausdr>) Das Drücken der Esc-Taste bricht die Eingabe über READ ab (ON).

SET EXACT ON | **OFF** Gilt ON, so liefert der Vergleich unterschiedlich langer Zeichenketten immer falsch (.F.).
SET EXCLUSIVE ON | **OFF** | (<l_ausdr>) Exclusiver Dateizugriff.
SET FILTER TO [<bedingung>] Definition einer globalen Filterbedingung.
SET FIXED ON | **OFF** | (<l_ausdr>) Erzwingen der Ausgabe mit Dezimalstellen (ON).
SET FORMAT TO [<filename> | <proz_name>] Ein Formatfile (.fmt), ein Kommandofile (.prg) oder eine Prozedur wird für die Eingabe im Direktmodus aktiviert, so daß sie bei jedem READ abgearbeitet wird.
SET FUNCTION <n_ausdr> **TO** <c_ausdr>
Definition der Funktionstaste <n_ausdr>.
SET INDEX TO [<fileliste> | (<n_ausdr_liste>)] Eröffnen der aufgeführten Indexfiles.
SET INTENSITY ON | **OFF** | (<l_ausdr>) Eingabebereiche bei der Eingabe durch READ zwischen direkter und Standardfarbe umschalten (SET COLOR).
SET KEY <n_ausdr> **TO** [<proz_name>] Verbinden einer Taste mit einer Prozedur.
SET MARGIN TO <n_ausdr> Einstellen des linken Randes auf dem Drucker.
SET MESSAGE TO [<n_ausdr>[**CENTER** | **CENTRE**]] Positionierung der Texte bei @ ... PROMPT
SET ORDER TO <n_ausdr> Das Indexfile mit der Ordnungszahl n_ausdr in der aktuellen Indexfileliste wird zum Hauptindex (0 ... 15).
SET PATH TO [<pfadliste>| (<c_ausdr>)] Definition eines Suchpfades.
SET PRINT ON | **OFF** | (<l_ausdr>) Ausgabe auf den Drucker (außer @-Ausgaben).
SET PRINTER TO [<gerät> | <file> | | (<c_ausdr>)] Auswahl eines Zielgerätes für die Druckerausgabe.
SET PROCEDURE TO [<proz_file>] Eröffnen eines Prozedurfiles.
SET RELATION [ADDITIVE] TO
[<schlüssel_1> | <satznr_1> **INTO** <alias_1> | (<c_ausdr_1>)
{, **TO** <schlüssel_i> | <satznr_i> **INTO** <alias_i> | (<c_ausdr_i>) ...}]
Logische Verbindung eines Datenbankfiles mit anderen Datenbankfiles über Indexschlüssel oder Satznummer.
SET SCOREBOARD ON | **OFF** | (<l_ausdr>) Anzeige (ON) der Mitteilungen von READ und MEMOEDIT in Zeile 0.
SET SOFTSEEK ON | **OFF** | (<l_ausdr>) Positionierung des Datensatzzeigers bei erfolglosem SEEK auf den Satz, der dem Schlüssel folgt (ON).
SET TYPEAHEAD TO <n_ausdr> Größe des Tastaturpuffers festlegen.
SET UNIQUE ON | **OFF** | (<l_ausdr>) Der erste (ON) oder alle Datensätze mit gleichem Schlüssel werden beim Indizieren in die Indexfiles aufgenommen.
SET WRAP ON | **OFF** | (<l_ausdr>) ON bewirkt in @ ... PROMPT-Menüs eine zyklische Bewegung des Kursors.
SKIP [<n_ausdr>] [**ALIAS** <alias>]
Im Arbeitsbereich <alias> wird der Datensatzzeiger relativ zur Ausgangsposition und entsprechend der logischen Ordnung des Files bewegt.
SORT TO <zielfile> | (<c_ausdr>) **ON** <feld_1>] [**/A** | **/D** | **/C**]
{,<feld_i>] [**/A** | **/D** | **/C**] ...} [<gb>] [**FOR** | **WHILE** <bedingung>]
Das aktive Datenbankfile wird nach den angegebenen Feldern und der angegebenen Reihenfolge sortiert.
STORE <ausdr> **TO** <var_liste>
<speichervar> = <ausdr>
Speichern des Wertes des Ausdrucks in alle angegebenen Speichervariablen.
SUM [<gb>] <ausdr_liste> **TO** <var_liste> [**FOR** | **WHILE** <bedingung>]
Berechnen der Summen der in der Ausdrucksliste angegebenen Werte für alle ausgewählten Datensätze.

TEXT [**TO PRINT** | **TO FILE** <filename> | (<c_ausdr>)]
<text>
ENDTEXT
Ausgabe eines Textes ohne Makroersetzung.
TO Siehe @ ... TO.
TOTAL ON <schlüssel> **TO** <filename> | (<c_ausdr>) [<gb>]
[**FIELDS** <feldliste>] [**FOR** | **WHILE** <bedingung>]
Komprimieren eines Datenbankfiles, wobei alle Sätze mit gleichem Schlüssel zu einem Satz zusammengefaßt werden. Die in der Feldliste aufgeführten numerischen Felder werden addiert.
TYPE <filename> | (<c_ausdr_1>) [**TO PRINT**] |
[**TO FILE** <filename> | (<c_ausdr_2>] Ausgabe eines Textfiles.

UNLOCK [**ALL**] Zugriffsschutz für Files und Sätze aufheben.
UPDATE [**RANDOM**] **ON** <schlüssel> **FROM** <alias> | (<c_ausdr>)
REPLACE <feld_1> **WITH** <ausdr_1> {,<feld_i> **WITH** <ausdr_i>...}
Korrektur eines Datenbankfiles, wobei die Korrekturdaten von einem anderen File eingelesen werden können.
USE [<filename> | (<c_ausdr_1>)] [**INDEX** <fileliste>]
[**ALIAS** <alias>] [**EXCLUSIVE**]
Das Datenbankfile <filename> wird mit den in der <fileliste> aufgeführten Indexdateien in Benutzung genommen.

WAIT [<prompt>] [**TO** <speichervar>] Die Abarbeitung des Programms wird unterbrochen, bis eine Taste gedrückt wird. Deren Zeichenkode wird der Speichervariablen zugewiesen.

ZAP Löschen aller Datensätze des aktiven Datenbankfiles.

Die Clipper-Funktionen folgen im Heft 4/1990.

zusammengestellt von Dr. Wilfried Grafik

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
4. Jahrgang · ISSN 0232-2892

Mikroprozessortechnik, Heft 3 · 90

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR-1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Hans Weiß, Verantwortlicher Redakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2870203), Hans-Joachim Hill (Tel. 2870209); Sekretariat Tel. 2870381

Gestaltung Christina Bauer

Titelbild Stefan Seebold, Thomas Neumann

Beirat Dr. Ludwig Claßen, Dr. Heinz Florin, Prof. Dr. sc. Rolf Giesecke, Joachim Hahne, Prof. Dr. sc. Dieter Hammer, Prof. Dr. sc. Thomas Horn, Prof. Dr. Albert Jugel, Prof. Dr. Bernd Junghans, Dr. Dietmar Keller, Prof. Dr. sc. Gernot Meyer, Prof. Dr. sc. Bernd-Georg Münzer, Prof. Dr. sc. Peter Neubert, Prof. Dr. sc. Rudolf Arthur Pose, Prof. Dr. sc. Dr. Michael Roth (Vorsitzender), Dr. Gerhard Schulze, Prof. Dr. sc. Manfred Seifart, Dr. Dieter Simon, Dr. Rolf Wätzig, Prof. Dr. sc. Dr. Jürgen Zaremba

Lizenz-Nr. 1710 des Presse- und Informationsdienstes der Regierung der DDR

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

Redaktionsschluß 9. Januar 1990

AN (EDV) 49837

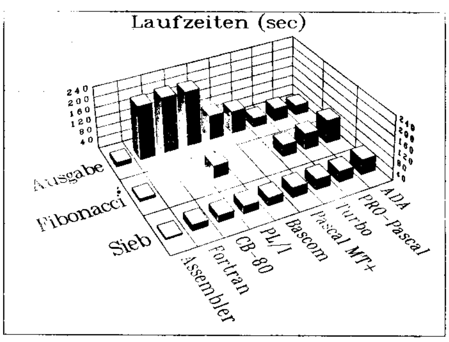
Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *Republik Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Republik Ungarn:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nôi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR-7010 Leipzig und Leipzig Book Service, Talstraße 29, DDR-7010 Leipzig

Die Bezeichnung Clipper steht im Englischen unter anderem für Schnellsegler und Transozeanflugzeug. Vermutlich kommt in der nächsten Zeit eine weitere Bedeutung hinzu: schneller Compiler für eine Datenbanksprache. Auf der **Seite 67** stellen wir Ihnen die Version Sommer '87 im Überblick vor und ergänzen den Beitrag mit einer Referenzkarte auf der **2. und 3. Umschlagseite**, die alle Befehls- und Schlüsselwörter enthält und Ihnen so den Einstieg und die Arbeit mit Clipper, dem erfolgreichen Compiler der Firma Nantucket, erleichtern soll. Die Funktionen von Clipper folgen im Heft 4/1990.

Im Test:



Auf der **Seite 74** finden Sie einen Testbericht zu Compilern, Interpretern und zu einem Z 80-Assembler unter CP/M. Dieser Beitrag stellt eine Weiterführung des Beitrages zu C-Compilern unter CP/M in MP 1/90 dar. Beide Tests sollen die Auswahl einer Sprache für die Lösung bestimmter Probleme erleichtern helfen.

Vorschau

Für das Heft 4/1990 bereiten wir für Sie unter anderem Beiträge zu folgenden Themen vor:
- Der Cache
- Im Test: Programmiersprachen für MS-DOS
- Lernmethoden der Künstlichen Intelligenz
- Mikroprozessorsteuerungen

Inhalt

Referenzkarte Clipper 2. US

MP-Dialog 66

Wilfried Grafik:
Clipper im Überblick 67

Michael Krapp, Volker Zerbe, Detlef Jahn:
Systemprogrammierung in Modula-2 69

Rainer Ploss:
Kodierung von Wordstar-Files 71

Peter E. Nebe:
Neuer Funktionsinterpreter für Turbo-Pascal 72

Uwe Schulze:
Im Test: Programmiersprachen für CP/M 74

Ralf Bergmann, Mathias Weidner:
Die serielle Schnittstelle unter DOS 75

Detlef Grell:
Systeme mit System (Teil 1) 78

MP-Kurs: 79
Manfred Zander:
Turbo-Pascal-Praxis

Michael Bach, Karl-Friedrich Probst:
UeberAll das gleiche 86

Christian Hanisch:
Schreibschutz per Software 88

Olaf Ginter:
Prozedurrufe zurückverfolgen 90

MP-Computer-Club 91

Dietmar Gralla:
Strukturiert programmieren in Basic

Torsten Schmidt, Marco Thäle:
Plottersimulation mit KC 85/3 in Forth

MP-Börse 93

MP-Bericht 94
INTERKAMA 89

Entwicklungen und Tendenzen 95

Verbindung aufnehmen 96

vorgestellt 4. US
Multiprozessor-PC Compaq Systempro

Liebe Leserinnen und Leser,

im vorigen Heft informierten wir Sie bereits über eine Vereinbarung zwischen unserer Redaktion und der Redaktion c't, die den gegenseitigen Nachdruck von Beiträgen regelt.
Die seit 1984 im Verlag Heinz Heise erscheinende c't kann nicht zuletzt dank der Qualität der Beiträge heute eine monatliche Verkaufsauflage von etwa 105 000 Exemplaren vorweisen und wird als führend unter den technisch orientierten Computerfachzeitschriften der BRD angesehen. Auch sie wurde bei Gründung der MP von Lesern hinsichtlich des Profils verschiedentlich als Vorbild genannt (wobei leider allzuoft die gravierenden Unterschiede in den jeweiligen Medienlandschaften und den äußeren Bedingungen übersehen wurden); so daß Artikel aus der c't zweifellos ein nützlicher Beitrag sind, um vorhandenes Know-how den Lesern in unserem Lande zugänglich zu machen.
Für dieses Heft haben wir als erstes einen Überblicksartikel ausgewählt – Systeme mit System –, der mit dem Aufzeigen des Entwicklungsweges des IBM-PCs und der Kompatiblen auch Kriterien der PC-Auswahl bietet. Der zweite Beitrag stellt ein kleines Programm zum Abtippen vor, mit dem DOS-Kommandos wie Dir oder Erase im aktuellen Verzeichnis und in dessen Unterverzeichnissen ausgeführt werden können. (Nicht übersehen sollten Sie auch die Ost-West-Kontaktbörse auf den Seiten 95 und 96, eine Gemeinschaftsaktion mit der c't, die wir noch nach Redaktionsschluß ins Heft nehmen konnten.)
Daraus ist zu ersehen, daß wir uns bei der Auswahl der c't-Artikel vom gleichen Prinzip leiten lassen, das für die MP insgesamt gilt: Vielfalt. Das bedeutet eine Mischung aus leichtverständlichen Überblicks- und Einführungsartikeln und „knackigen" Tips und Tricks mit komplettem Listing oder einer Schaltung zum unmittelbaren Nachnutzen. Wenn es um internationale Trends und Tendenzen der Computerbranche geht, werden wir uns um höchste Aktualität bemühen; Wissensvermittlung als Hilfe bei der praktischen Arbeit wird sich vor allem an den in unserem Lande vorhandenen Gegebenheiten orientieren. Die Erstveröffentlichung darf dann wohl auch einmal etwas weiter zurückliegen.
Unsere besonders aufmerksamen Leser bitten wir um Verständnis dafür, daß es – aus technischen Gründen – bei einigen Begriffen noch zu Abweichungen gegenüber der in der MP üblichen Schreibweise kommen kann; beispielsweise bei MS-DOS oder Basic. Da diese Abweichungen jedoch unbedeutender Natur sind, kann sie der Interessierte als kleine Schönheitsfehler sicherlich verschmerzen. Aus den genannten Gründen muß hin und wieder auch auf Fotos – insbesondere Farbfotos – verzichtet werden, die zur Verständlichkeit eines Beitrages nicht dringend notwendig sind.
Insgesamt meinen wir, daß dieser unser erster Schritt zur Kooperation mit einem westlichen Partner die Zustimmung der Mehrheit unserer Leser finden wird; wir würden uns aber dennoch freuen, Ihre Meinungen zu erfahren. In diesem Sinne: Postkarte oder Anruf genügt!

Ihre Redaktion MP

Zum Beitrag „Logischer Entwurf von Datenbanken"

aus MP 7/89

Mit diesem Beitrag wird ein wichtiges, oft unterschätztes Problem aufgegriffen. Gerade auch deshalb sind aus meiner Sicht einige kritische Bemerkungen und Ergänzungen unerläßlich, wobei ich aus Platzgründen leider nicht auf Einzelheiten eingehen kann.
Insgesamt ist einzuschätzen, daß die Problematik des logischen Entwurfs von Datenbanken unzureichend, unexakt und fehlerbehaftet dargestellt ist.
Zum Verständnis dringend notwendiges Kontextwissen (relationales Konzept, Verhältnis Großcomputer-DBMS – Mikrocomputer-DBMS) wird nicht vermittelt. Verschiedene Begriffe werden nicht exakt verwendet bzw. verwechselt (Datei – Datenbank, Vermischung der logisch orientierten mit der speichertechnisch geprägten Terminologie). Statt des üblichen Begriffs „Normalform" wird überflüssigerweise „Strukturform" verwendet.
Die Motivation für die Überprüfung einer Datenstruktur in die 1. Normalform wird unzureichend, die 2. Normalform wird schlicht falsch erklärt. Die Erläuterungen zur 3. Normalform treffen nicht das Wesentliche, da die gezeigten Redundanzen immer auf die beschriebene Art beseitigt werden können, unabhängig von der Betrachtung der Normalformen. Genauer und umfassender kann sich der Leser zu diesem Problemkreis in der im Beitrag zitierten Broschüre bzw. der darin aufgeführten Literatur informieren.
Die in Bild 5 dargestellte Datenstruktur ist zwar nicht falsch, aber unter dem Gesichtspunkt einer Mikrocomputeranwendung (die eingangs explizit betont wird) völlig unrealistisch. Dies begründet sich, kurz gesagt, mit unzumutbar hohen Abarbeitungszeiten, die infolge der Aufsplittung der Datenstruktur in eine Vielzahl von Dateien entstehen. Dieser Sachverhalt ist Ausdruck einer immer wieder auftretenden Fragestellung, die jeweils in Abhängigkeit vom Anwendungsfall entschieden werden muß, und zwar unter Berücksichtigung der Anforderungen an die Datenbanklösung. Dabei geht es wesentlich um das Verhältnis von Update und Retrieval, welches über den Grad der Aufsplittung bzw. über die strenge Umsetzung der Normalformenlehre entscheidet. Oftmals ist eine Priorität des Retrieval zu verzeichnen, was in der Regel einen Verzicht auf die angestrebte 3. Normalform (manchmal sogar die 1. Normalform) bedingt. Kurz gesagt: Die Normalformen sind nicht als Dogma zu betrachten, sondern als Entwurfsrichtlinien, die allerdings bei Mikrocomputeranwendungen wegen der resultierenden zu langen Abarbeitungszeiten oft nicht konsequent berücksichtigt werden können. Trotzdem ist die Auseinandersetzung mit diesen Normalformen zu empfehlen, da sie eine Abkehr von der meist empirischen Herangehensweise beim Datenbankentwurf unterstützen.

Dr. D. Langbein

Dieser Kurs ist richtig

Der Kurs Turbo-Pascal-Praxis fand bei unseren Lesern großen Anklang, und so erhielten wir auch Verbesserungsvorschläge zu einigen Programmlösungen und Angebote, einzelne Themen ausführlicher darzustellen. Wir bedanken uns auf diesem Wege für alle Zuschriften, können aber aus Platzgründen nicht näher darauf eingehen. Durch den Kurs konnten wir aber offensichtlich auch erreichen, daß einige PC-Nutzer die Sprache Turbo-Pascal für sich neu oder überhaupt erst entdeckten.
So fand Peter Busch aus Johanngeorgenstadt „angeregt durch ... (den) Kurs ‚Turbo-Pascal-Praxis' rasch großes Interesse an dieser Programmiersprache. Der Kurs ist sehr anschaulich und leicht verständlich, so daß auch mir als ... Anfänger ... der Einstieg recht schnell gelang." Dieses Lob haben wir gerne an Herrn Zander weitergeleitet und hoffen, daß sich noch mehr Autoren als bisher um einen solchen leichtverständlichen Schreibstil bemühen.
Um noch einmal auf Herrn Busch zurückzukommen, er nutzte nicht nur den Kurs, sondern entwickelte selbst eine kleine Prozedur, die wir Ihnen nicht vorenthalten wollen, weniger um damit Profis hinter dem Ofen hervorzulocken, als mehr den Unentschlossenen unter Ihnen noch eine Anregung zu geben, sich dieser interessanten und vielseitigen Programmiersprache zu widmen.

Redaktion MP

```
program feiertag;
uses crt,dos;              {Turbo-Pascal 5.0}

procedure feiertage;
{ Gültig für die Jahre 1900 bis 2099 }
var i,o,p:integer; m:string[5];
begin
    clrScr;
    gotoxy(30,10); write('Jahr : '); read(i);
    o:=(204-11*(i mod 19)) mod 30;
    case o of 28,29:o:=pred(o); end;
    o:=28+o-(i+i div 4+o-13) mod 7;
    if o>31 then begin o:=o-31; m:='April '; end
                 else m:='März ';
    gotoxy(30,13); write('Ostersonntag  : ');
    write(o:2,'. '+m,' ',i,'                      ');
    if m='März ' then begin p:=o-11; m:='Mai  '; end
                 else if o<12 then begin p:=o+20; m:='April '; end
                          else begin p:=o-11; m:='Juni  '; end;
    gotoxy(30,15); write('Pfingstmontag : ');
    writeln(p:2,'. '+m,' ',i,'                      '); writeln;
end;

begin
  feiertage; delay(5000)
end.
```

Clipper im Überblick

Dr. Wilfried Grafik, Berlin

Clipper ist ein Programm, mit dem in Clipper, dBase III, dBase III Plus oder auch Foxbase geschriebene Kommandofiles compiliert werden können. Im engeren Sinne handelt es sich also um einen Compiler. Im weiteren Sinne versteht man darunter eine Sammlung von Programmen, die eine Arbeitsumgebung für diesen Clipper-Compiler bilden. Dieser Beitrag beantwortet die häufigsten Fragen, die beim Ersteinsatz von oder beim Umstieg auf Clipper auftreten. Die Referenztabelle auf der 2. und 3. Umschlagseite faßt die Syntax der Kommandos zusammen, die Funktionen folgen im Heft 4.

Interpreter und Compiler

Die Programme dBase III und dBase III Plus sind Interpreter. Sie akzeptieren eine Anfragesprache (query language), die aus allen Kommandos besteht, die im Dialog von der Tastatur aus eingegeben werden können. Außerdem können Kommandofolgen in Kommandofiles gespeichert und abgearbeitet werden. Das wesentliche Merkmal eines Interpreters besteht jedoch darin, daß er eine Einheit der Eingabesprache, hier eine Zeile, analysiert, sie anschließend ausführt, zur logisch nächsten Zeile geht, diese analysiert, ausführt usw. Werden Zeilen mehrfach abgearbeitet, etwa in DO WHILE-Zyklen, so werden diese Zeilen auch mehrfach analysiert, was eigentlich nicht nötig wäre, da ihre Korrektheit bereits durch die erste Analyse festgestellt wurde. In dieser wiederholten Analyse besteht ein entscheidender Nachteil von Interpretern, der die Abarbeitungsgeschwindigkeit interpretierter Programme erheblich senkt. Der Vorteil eines Interpreters besteht darin, daß die unmittelbare Verbindung von Quelltext und intern ausgeführten Anweisungen erhalten bleibt. Interpreter eignen sich deshalb sehr gut für die interaktive Arbeit und zum Test von Programmen.
Ein Compiler ist dagegen ein Programm, welches eine Quellsprache, hier die Kommandosprache von Clipper, in eine Zielsprache übersetzt. Zielsprache des Clipper-Compilers ist die Maschinensprache. Genauer gesagt: Bei der Übersetzung wird ein Objektmodul (Namenserweiterung .obj) erzeugt, der im Speicher verschieblich ist. Seine endgültige Lage im Speicher kann erst dann bestimmt werden, wenn alle Objektmodule, die zu einem Programm gehören, verfügbar sind und zu einem ausführbaren File zusammengefügt werden (Namenserweiterung .exe). Diesen Vorgang bezeichnet man als Linken (verbinden). Da zur Ausführung verschiedener Programme gleiche Routinen benötigt werden, etwa zum Sortieren, werden diese vorübersetzt und in einer Bibliothek zusammengestellt (clipper.lib) Aus dieser Bibliothek holt sich der Linker die für ein Programm erforderlichen Objektmodule. Sie muß dem Linker deshalb als zu benutzende Bibliothek angegeben werden.
Die Erstellung eines Clipper-Programms besteht also aus drei Schritten: der Eingabe der Kommandofiles mit einem Editor, der Übersetzung der einzelnen Kommandofiles und dem Linken.

Anforderungen und Grenzen

Clipper und dBase III (Plus) stellen die gleichen Anforderungen an die Hard- und Software, das heißt, ein Rechner, auf dem dBase III läuft, ist auch für Clipper geeignet (XT oder AT, 256 KByte Hauptspeicher, ein Laufwerk). Ob ein konkretes Projekt mit diesen Minimalanforderungen auskommt, ist allerdings fraglich. Möchte man eine akzeptable Arbeitsumgebung für Clipper zusammenstellen, so muß mit einem Festplattenbedarf von etwa 1,5 MByte gerechnet werden, zuzüglich der Anwenderprogramme, Datenbankfiles, Editor usw. Für ein realistisches Projekt sollte deshalb unbedingt ein Rechner mit Festplatte verwendet werden.
Die internen Grenzen von Clipper sind im Vergleich zu dBase III erweitert. Die markantesten Änderungen sind in Tafel 1 zusammengestellt.

Tafel 1

	Clipper	dBase III
Zeichen je Zeichenfeld	≦ 32KByte	≦ 254
Zeichen je Momo-Feld im .dbt-File	≦ 64KByte	≦ 4096
Zeichen je Satz	begrenzt durch RAM	≦ 4000
Felder je Satz	≦ 1024	≦ 128
aktive Speichervariablen	≦ 2048	≦ 256
Prozeduren je Prozedurfile	≦ 255	≦ 32
gleichzeitig eröffnete Files	≦ 255	≦ 15
gleichzeitig eröffnete Indexfiles	≦ 15	≦ 7
Länge eines Indexschlüssels	≦ 250	≦ 100

Erweitert ist ebenfalls die Anzahl der mit einem Datenbankfile verbundenen Datenbankfiles (SET RELATION). Während dBase III und dBase III Plus von einem Datenbankfile aus nur eine Verbindung zulassen, gestattet Clipper bis zu acht solcher Verbindungen (Bild 1).

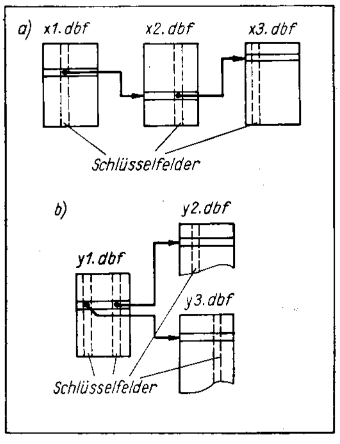


***Bild 1** Verbindungsstrukturen zwischen Datenbankfiles (SET RELATION)*
***a)** dBase III und dBase III Plus*
***b)** Clipper*

Übernahme von Datenbanken

Datenbankfiles (.dbf), einschließlich der zugehörigen Memo-Files (.mem), aus dBase III, dBase III Plus oder FoxBase werden von Clipper akzeptiert. Umgekehrt akzeptieren diese Interpreterversionen auch unter Clipper erstellte Datenbankfiles, selbst wenn diese mehr als 4000 Zeichen je Datensatz enthalten.
Bildschirmmasken (.fmt), Berichtsformate (.frm), Adreßformate (.lbl) und Memoryfiles (.mem) aus dBase III werden von Clipper akzeptiert. Indexfiles von dBase III und dBase III Plus (.ndx) können übernommen werden, obwohl Clipper eine eigene Verwaltung seiner Indexfiles besitzt. Neben einer anderen Namenserweiterung (.ntx) ist auch ihr interner Aufbau anders. Sollen Indexfiles von dBase III verwendet werden, so liegt der einzige Unterschied zur Verwendung Clipper-eigener Indexfiles beim Linken: Den zu verbindenen Modulen wird der Standardmodul ndx.obj hinzugefügt.
Die Übernahme der Kommandofiles bedarf einer genaueren Betrachtung, die nachfolgend in groben Zügen erfolgen soll. Prinzipiell kann festgestellt werden, daß Clipper alle Kommandos aus dBase III akzeptiert, die nicht unmittelbar oder mittelbar die Arbeit im Direktmodus betreffen. Zusätzlich gibt es aber eine Fülle neuer, leistungsfähiger Kommandos und Funktionen, die der Nutzer bei einiger Erfahrung im Umgang mit Clipper nicht mehr missen möchte. Damit erscheint ein ständiger Wechsel zwischen dBase und Clipper nicht sehr vorteilhaft.

Vorteile

Neben den erweiterten Grenzen gegenüber dBase III besitzt Clipper eine Reihe von Vorteilen, die in ihrer Gesamtheit erst durch die praktische Anwendung deutlich werden. Sie hängen unter anderem auch von der Zielstellung konkreter Projekte ab. So kann für ein kleines Projekt mit viel Dateiarbeit und häufigen Veränderungen durch dBase III-kundige Nutzer eine Interpreterversion durchaus geeignet sein. Die meisten Nutzer kommen über dBase III zu Clipper, wenn ihre Datenbankfiles zu groß werden (über 3000 Datensätze), die Bildschirmarbeit zu langsam wird oder eine akzeptable Nutzeroberfläche mit dBase III nicht mehr programmierbar ist. Weitere Entwicklungen werden dann gleich in Clipper geschrieben, da neue Projekte, die von vornherein mit Clipper konzipiert werden, eine höhere Qualität im Vergleich zu dBase III-Projekten haben. Im folgenden betrachten wir einige dieser Vorteile.
Durch die Kompilation steigt die Abarbeitungsgeschwindigkeit, insbesondere für Operationen im Speicher und mit dem Bildschirm, erheblich. Dazu ein Beispiel: Das Kommando @... BOX aus Clipper wurde unter dBase III als Prozedur nachgebildet. Baut man mit dieser BOX-Prozedur das erste Menübild von SET unter dBase III auf, so werden auf einem XT dafür etwa 40 Sekunden benötigt. Ein Clipper-Programm baut das gleiche Menü in weniger als 2 Sekunden auf. Ungeachtet der Tatsache, daß in dBase III und dBase III Plus durch geschicktes Programmieren für diesen speziellen Fall auch etwas kürzere Zeiten erreichbar sind, beispielsweise durch Benutzung von SPACE, sind auch Fortschritte bei den einzelnen Clipper-Versionen zu verzeichnen. Auf einem AT-Rechner können mit Clipper Menübilder, einschließlich der Bewegung eines Balkenkursors, erstellt werden, die praktisch *sofort* erscheinen. Erst diese Qualität gestattet den Aufbau komfortabler Nutzeroberflächen.
Ein weiterer Vorteil ist darin zu sehen, daß fertige Clipper-Projekte vom Endnutzer nicht veränderbar sind, da nur ausführbare Files ausgeliefert werden. Die Quellprogramme verbleiben beim Entwickler. Für mehrfach genutzte Projekte, beispielsweise in ver-

schiedenen Betriebsteilen, bleibt dadurch die Kompatibilität erhalten.

Erweiterungen

Clipper enthält kompakte Kommandos zum Bildschirmaufbau (@ ... BOX, @ ... TO), zum Menüaufbau (SET MESSAGE TO, MENUE TO, @ ... PROMPT), zum Retten und Wiederherstellen von Bildschirminhalten (SAVE SCREEN, RESTORE SCREEN), Funktionen zur Abfrage der Tastatur (INKEY(), LASTKEY(), NEXTKEY()) und zur Fensterarbeit (SAVESCREEN (...), RESTSCREEN (...)), die sehr kurze Abarbeitungszeiten haben. Sie tragen wesentlich zur Akzeptanz von Clipper durch den Nutzer bei. In den verschiedenen Clipper-Versionen wurden sie ständig erweitert und beschleunigt. Die Grundfunktionen von Clipper und das Laufzeitsystem sind in der Bibliothek *Clipper.lib* enthalten. Zusätzliche Funktionen befinden sich in der Bibliothek *Extend.lib*. Eng damit verbunden sind auf den Endnutzer orientierte (private) Hilfen. Sie werden in zwei Gruppen eingeteilt: kontextbezogene und allgemeine Hilfen. Kontextbezogene Hilfen können immer dann aktiviert werden, wenn eine Eingabe von der Tastatur über READ angefordert wird, der Zusammenhang, in dem die Hilfe erfolgen soll, also genau feststeht. Durch Drücken der [F1]-Taste wird an den Anfang des Prozedurfiles *Help.prg* verzweigt, wobei drei Parameter vom Laufzeitsystem übergeben werden: Name der Prozedur, in der [F1] gedrückt wurde, die Zeilennummer im Kommandofile (nicht in der Prozedur!) und der Name der Variablen, für die eine Eingabe erwartet wird. Die erste Zeile des Prozedurfiles *Help.prg* ist deshalb das Kommando PARAMETERS:

PARAMETERS calledBy, fromLine, varName

Das Kommandofile *Help.prg* muß vom Anwender geschrieben und mit dem Hauptprogramm durch Linken verbunden werden. Durch Auswertung der Parameter, gegebenenfalls nur durch deren Ausgabe, wird eine geeignete Hilfestellung programmiert, etwa das Retten des Bildschirminhalts (SAVE SCREEN TO) und die Ausgabe eines Kastens (@ ... BOX) mit entsprechendem Inhalt.

Allgemeine Hilfen können zu jeder Zeit des Programmlaufs erreicht werden. Auch hier muß ein Prozedurfile *Help.prg* mit dem Hauptprogramm verbunden sein. Für den Informationsaustausch zwischen Programm und Helpfile wird die Benutzung globaler Variabler empfohlen.

Clipper gestattet es dem Nutzer, Funktionen zu definieren. Eine Funktion kann als eine Prozedur aufgefaßt werden, die einen bestimmten Ergebniswert liefert. Syntaktisch unterscheiden sich Prozedur- und Funktionsdefinitionen durch das einleitende Schlüsselwort (PROCEDURE, FUNCTION) und durch ihre Rückkehranweisung. Funktionen erfordern die Rückkehr über RETURN *mit* einem nachfolgenden Ausdruck. Die Berechnung dieses Ausdrucks liefert den Ergebniswert der Funktion. Beim Übersetzen überprüft Clipper, ob PROCEDURE und RETURN bzw. FUNCTION und RETURN <ausdr> zusammenpassen. Fehlermeldungen erscheinen, wenn bei der Rückkehr aus einer Prozedur ein Ergebniswert angegeben ist und wenn eine Funktion durch RETURN ohne Ergebniswert beendet wird. Fehlt RETURN <ausdr> am Ende einer Funktionsdefinition allerdings völlig, so wird kein Fehler gemeldet. Tests ergaben, daß der Ergebniswert **.F.** geliefert wird. Die Typverträglichkeit der zurückgegebenen Werte wird nicht überprüft. Der Aufruf einer vom Nutzer definierten Funktion ohne Parameter muß, wie auch bei den Standardfunktionen, immer mit Parameterklammern erfolgen.

Da dBase III keine Felder kennt, müssen sie unter Ausnutzung der Makrotechnik mühselig nachgebildet werden. Als Erweiterung wurden in Clipper deshalb eindimensionale Felder aufgenommen. Ihre Vereinbarung wird mit dem Schlüsselwort DECLARE eingeleitet.

DECLARE a[3]

vereinbart beispielsweise ein Feld mit dem Namen **a**, das aus drei Elementen besteht. Diese werden durch a[1], a[2] und a[3] erreicht. Die untere Indexgrenze ist immer 1. Während des Laufes erfolgt ein Test auf Einhaltung dieser Indexgrenzen. Wird ein Indexwert berechnet, der außerhalb dieser Grenzen liegt, so erfolgt eine Fehlermitteilung (subscript range). Felder in Clipper haben nur wenig Ähnlichkeit mit Feldern in bekannten höheren Programmiersprachen wie Pascal, Modula oder C. Ihre Feldelemente müssen nicht vom gleichen Typ sein. Die Bibliothek *Extend.lib* enthält unter anderem Prozeduren und Funktionen zur Manipulation von Feldern.

Im Zusammenhang mit der Deklaration von Feldern ist die Aufnahme des neuen Kommandos FOR ... NEXT zu sehen. Es dient in erster Linie der zyklischen Behandlung von Feldelementen. Mehrdimensionale Felder müssen auch in Clipper durch Makros nachgebildet werden.

Durch die Übersetzung mit Clipper fehlt dem Nutzer die Dialogebene von dBase III (obwohl es Nachbildungen der Dialogebene als Prg-Files gibt), insbesondere der Bezug zu den Zeilen und Bezeichnern in den Kommando- und Datenbankfiles. Für inhaltsreiche Fehlernachrichten ist der Zugriff auf diese Informationen wichtig. Deshalb wurden die zusätzlichen Funktionen FIELDNAME(), PROCLINE() und PROCNAME() aufgenommen. Weitere Funktionen dienen dem Aufbau eines Nutzerdialogs, beispielsweise dem Zugriff auf Memofelder (MEMOEDIT, MEMOREAD, MEMOTRAN), was in dBase III ausschließlich auf der Dialogebene erfolgt. Auch bezüglich der Arbeit im Direktmodus stellt die Bibliothek *Extend.lib* einige Prozeduren bereit.

Unterschiede

Im Vergleich zu dBase III und dBase III Plus existieren einige syntaktische und semantische Unterschiede, die nicht immer offensichtlich sind und bei der Überführung beachtet werden müssen. So bewirkt das Weglassen der Ausdrucksliste bei DISPLAY oder LIST nicht die Ausgabe von Feldinhalten. Fehlt OFF in diesen Kommandos, so werden nur die Satznummern ausgegeben, mit OFF leere Sätze.

Die Kommandos AVERAGE und SUM verlangen eine Variablenliste, da ansonsten unklar wäre, was mit den berechneten Werten geschehen soll. Würde eine Ausgabe auf dem Bildschirm erfolgen, wie in dBase III, so wäre es im allgemeinen schwer, den Sinn dieser Ausgaben zu erfassen.

Erweitert wurde die Semantik des CALL-Kommandos und der Parameterübergabe. CALL kann für den Aufruf *beliebiger* Routinen, die in *Fremdsprachen* (C, Assembler) geschrieben sind, verwendet werden. Diese müssen beim Linken mit dem Clipper-Programm verbunden werden. Das trifft insbesondere für in *Fremdsprachen* geschriebene Routinen zu, die beispielsweise in einer Bibliothek abgelegt sein können. So können einige Passagen im Clipper-Programm noch effektiver geschrieben werden, was allerdings mit einer verminderten Portabilität solcher Programme bezahlt werden muß.

Mit CALL können keine in Clipper geschriebenen Programme gerufen werden. CALL nutzt für die Parameterübergabe, die in C-Systemen auf XT-/AT-Rechnern gebräuchliche Verfahrensweise. Diese Konventionen zur Parameterübergabe in Clipper und in einigen C-Implementationen sind aber unterschiedlich. In der gerufenen Prozedur müssen alle Parameter als Referenzparameter vereinbart sein (Pointer). Für eine Zeichenkette wird die Adresse des ersten Bytes übergeben. Die Zeichenkette selbst wird durch ein Nullbyte abgeschlossen. Für numerische Werte wird ein Pointer auf eine Gleitkommazahl übergeben. Möchte man einen ganzzahligen Typ erzwingen, so ist im Clipper-Programm die Typkonvertierungsfunktion WORD anzuwenden.

Beispiel:

CALL far_proc WITH "string", 1, WORD (1)

Hierbei ist *Far_proc* der Name der gerufenen Prozedur, die drei Parameter hat. Die Typverträglichkeit und die Anzahl der Parameter in rufender und gerufener Prozedur können von Clipper nicht überprüft werden. Für den Aufruf in Clipper geschriebener Prozeduren ist die Verwendung von WORD sinnlos. Das Laufzeitsystem von Clipper erkennt beispielsweise in der Zeile

DO clipper_proc WITH WORD (1)

eine Typunverträglichkeit (type mismatch). Es sind je CALL-Prozedur höchstens 7 Parameter zugelassen.

Das Kommando PARAMETERS, das nur eine deklarative Wirkung hat, ist in Clipper auch für Kommandofiles zugelassen. Insbesondere kann es im Hauptmodul als erstes Kommando verwendet werden und gestattet dann einen Zugriff auf die Parameter der Kommandozeile beim Aufruf. Betrachten wir dazu als Beispiel das Kommandofile *Param.prg*:

```
* File param.prg
* Test der Parametübergabe aus der Kommandozeile
PARAMETERS par1, par2
? "1. Parameter:", par1
? "2. Parameter:", par2
RETURN
* eof param.prg
```

Nach dem Übersetzen und Linken rufen wir das Programm mit

c: >param abc 123

auf und erhalten dann:

1. Parameter: abc
2. Parameter: 123

Der Aufruf

c: >param abc,123

liefert dagegen die folgende Ausschrift:

1. Parameter: abc,123
2. Parameter:

und eine Fehlerausschrift in der ersten Bildschirmzeile (... undefined identifier PAR2).

Insgesamt sind die Fragen der Parameterübergabe, einschließlich der Typverträglichkeit und der Parameterarten (Werteparame-

ter, Referenzparameter), die Vermittlung von Ergebnissen nach außen, die Anpassung oder Neubestimmung von Typen, der Aufruf von und die Rückkehr aus fremdsprachigen Prozeduren derart umfangreich, daß sie einer gesonderten Betrachtung bedürfen. Ein Nutzer, der eine höhere Programmiersprache kennt, wird die ihm vertrauten Prinzipien in Clipper nicht durchgängig wiederfinden.

Ein weiterer Unterschied zu dBase III besteht in der Verwendung von Makros. Makroersetzungen in Clipper-Quelltexten dürfen keine Schlüsselwörter, jedoch Namen von Funktionen enthalten. Des weiteren sind solche Makroersetzungen verboten, die für die Übersetzung relevante Begrenzer enthalten. Diese Festlegung kann leider nicht exakter erfolgen, da sie nicht nur von den Zeichen an sich, sondern auch vom Kontext, in dem sie verwendet werden, abhängen. Zwei Beispiele sollen das verdeutlichen:

open = "USE myfile"
...
&open
Dies ist für Clipper falsch, da das Schlüsselwort USE ersetzt wird. Nach der Anweisung
feld = "a[3]"
ist
var = &feld
möglich, nicht aber
DECLARE &feld.
Die Makroersetzung in Clipper liegt dennoch auf einer anderen Ebene als Makroersetzungen in dBase, C oder Assemblersprachen.
Allgemein versteht man unter einem Makro eine Zeichenfolge, beispielsweise Befehle, die durch einen Namen bezeichnet sind. Die Zuordnung der Zeichenfolge zum Makronamen erfolgt in der Makrodefinition. In Clipper ist das die Zuweisung einer Zeichenkette zu einer Speichervariablen. Das Auftreten des Makronamens in einem Text bewirkt dann das Einfügen der durch diesen Namen definierten Zeichenkette an dieser Stelle des Textes. Diese Ersetzung erfolgt im allgemeinen auf Quelltextebene und vor der Analyse durch den Compiler oder Interpreter. In Clipper muß sie aber zur Abarbeitungszeit des übersetzten Programms erfolgen, das heißt, das Laufzeitsystem von Clipper muß die Ersetzung, Analyse, Übersetzung und Ausführung *während der Abarbeitung* des Nutzerprogramms durchführen.
Hingewiesen sei noch auf die veränderte Semantik von CREATE. CREATE <filename> erzeugt eine leere *Strukturdatei*. In diese leere Datei kann über APPEND BLANK und REPLACE die *Struktur* eines Datenbankfiles auch ohne Nutzerdialog eingegeben werden. Mit CREATE FROM wird dann das eigentliche Datenbankfile erzeugt. Zum Erstellen von Berichtsformaten oder Adreßformaten fehlen solche Kommandos. Dafür existieren in der Clipper-Umgebung zusätzliche Programme (.exe-Files).

Clipper-Umgebung

Zur Umgebung eines Compilers gehören mindestens ein Editor, ein Linker, ein Laufzeitsystem (Bibliothek) und ein Debugger. Weitere Servicefunktionen werden in sogenannten Toolboxen bereitgestellt. Clipper besitzt mit Ausnahme eines speziellen Editors alle diese Werkzeuge. Als Linker wird meist eine Version des PLINK86 empfohlen. Er ist im Vergleich zu anderen Linkern relativ langsam, gestattet aber dafür die Bildung von Überlagerungen. Für Clipper-Programme werden erfolgreich auch andere, schnellere Linker des MS-DOS eingesetzt. So kann ein übersetztes Programm (Myprog.obj), welches Indexfiles von dBase III (Ndx.obj) und Funktionen der *Extend.lib* benutzt, beispielsweise mit einem Linker des MSC-Systems (link) durch

link myprog+ndx,,nul,extend+clipper

verbunden werden. Die beiden aufeinanderfolgenden Kommas dienen als *Platzhalter* für die nicht benutzte Angabe des Ergebnisfilenamens. In diesem Beispiel entsteht ein ausführbares File mit dem Namen Myprog.exe. Die Bezeichnung *nul* gibt an, daß kein Map-File erstellt werden soll.
Neben diesen Hilfsmitteln gehören zur Clipper-Umgebung Programme, die einige Kommandos von dBase III nachbilden, die im Direktmodus arbeiten.

Systemprogrammierung in Modula-2

Dr. Michael Krapp, Volker Zerbe, Detlef Jahn, Ilmenau

In /1/ wurde die Programmiersprache Modula-2 bereits vorgestellt und auch eine geschichtliche Einordnung getroffen. Hervorgehoben sei nochmals, daß Modula-2 eine universelle Programmiersprache und besonders gut geeignet ist, Betriebssystemerweiterungen auf Hochsprachniveau programmieren zu können – Modula-2 ist also eine sogenannte Systemprogrammiersprache. Nachfolgend sollen die neuen Aspekte, welche Modula-2 in diesen Anwendungsbereich einbringt, genauer diskutiert und anhand von Programm-Fragmenten aus einem Multitasksystem veranschaulicht werden.

Modul-Konzept

Module sind Kompilationseinheiten und stellen Sammlungen von Konstanten, Variablen, Typen oder Prozeduren dar. Man unterscheidet im wesentlichen drei Modultypen: Hauptmodule, Definitionsmodule und Implementationsmodule. Ein Hauptmodul besteht aus einem Deklarationsteil und dem Ausführungsteil, ist gleichzeitig eine Programmeinheit und entspricht dem Hauptprogramm in konventionellen Programmiersprachen (z.B. program in Pascal). Der Hauptmodul kann mit Importlisten durch die Befehle **FROM <modulname> IMPORT <Leistungen>** auf Objekte anderer Module über ihre Definitionsmodule zugreifen. Er stellt aber nach außen eine geschlossene Einheit dar und exportiert keine Leistungen. Definitionsmodule und Implementationsmodule treten immer paarweise auf und sind durch gleiche Namen gekennzeichnet. Im weiteren wird mit *Modul* dieses Paar aus Definitions- und Implementationsmodul bezeichnet. Ein Definitionsmodul kann nur Deklarationen von Konstanten, Variablen, Typen oder Prozedurköpfen enthalten; ein Implementationsmodul dagegen kann neben Deklaration von Konstanten, Variablen und Typen auch vollständige Prozeduren enthalten. Wenn im Definitionsmodul Prozedurköpfe auftreten, müssen im Implementationsmodul diese Prozeduren vollständig deklariert sein. Alle sonstigen Deklarationen des Definitionsmoduls sind im zugehörigen Implementationsmodul verfügbar. Ein Definitionsmodul ist die Schnittstelle zur *Umwelt*, das heißt, alle Variablen, Konstanten, Typen oder Prozeduren, die im Definitionsmodul benannt sind, werden automatisch exportiert und somit dem Anwender zur Verfügung gestellt. Ein Implementationsmodul kann neben dem Deklarationsteil noch einen Ausführungsteil enthalten, der von den Schlüsselwörtern **BEGIN** und **END <modulname>** eingeschlossen sein muß, wobei die Ausführungsteile von importierten Modulen *vor* diesem Ausführungsteil ausgeführt werden. Dies gilt auch für den Ausführungsteil des Hauptmoduls. Durch diesen Mechanismus können die in einem Modul *verborgenen* Datenobjekte vor ihrer Benutzung initialisiert werden. Sowohl Definitionsmodul als auch Implementationsmodul können über Importlisten auf andere Definitionsmodule zugreifen.

Die Frage, was man in Definitions- und Implementationsmodulen deklariert, kann rein pragmatisch entschieden werden. So sind im Extremfall reine Datenstrukturmodule möglich, die nicht als Programmeinheiten interpretiert werden.
Da Importlisten auf Definitionsmodule verweisen, diese aber nur ganz bestimmte Teile von Deklarationen eines Moduls offenbaren (exportieren), ist die Möglichkeit des *Verbergens* von Implementierungsdetails gegeben. Damit ergeben sich qualitativ neuartige und weitreichende Abstraktionsmöglichkeiten beim Programmentwurf.
Außerdem resultieren daraus Möglichkeiten zur separaten Kompilation. Der Hauptmodul oder die Definitionsmodule können bereits übersetzt werden, wenn lediglich die von ihnen importierten Definitionsmodule übersetzt vorliegen. Ein Implementationsmodul kann dann übersetzt werden, wenn sein gleichnamiger Definitionsmodul und alle von ihm importierten Definitionsmodule übersetzt sind. Damit können Programmteile von verschiedenen Mitarbeitern bearbeitet und bei strenger Bindung an formale Schnittstellen gut entkoppelt werden, was bei komplexen Systementwicklungen bekannterweise eine praktische Notwendigkeit ist.
Abschließend sei erwähnt, daß es außer den genannten Modultypen noch den Typ des lokalen Moduls gibt. Dieser stellt keine Kompilationseinheit dar, da er in Haupt- oder Implementationsmodule eingebettet ist. In Erweiterung von lokalen Prozeduren kann man durch Exportlisten mit dem Befehl

EXPORT <Leistungen>;

oder durch Importlisten den Sichtbarkeitsbereich von Programmobjekten explizit steuern.

Koroutinen-Konzept

Als eine Verallgemeinerung des Prozedurkonzeptes (auch Unterprogramm- oder Routinen-Konzept) lassen sich mit diesem Konzept Programme quasiparallel (concurrent) ausführen. Eine Koroutine kann also ihren Ablauf unterbrechen und zwischenzeitlich den Befehlszähler einer anderen Koroutine überlassen. Diese Umschaltung erfolgt explizit. Im Unterschied zum Prozedurkonzept gibt es keinen Return-Mechanismus; eine Fortsetzung der unterbrochenen Koroutine ist daher nur über eine gleichartige explizite Umschaltung möglich. Alle Werte lokaler Datenobjekte bleiben während der Unterbrechungszeit erhalten. Damit sind Koroutinen nicht wie Prozeduren hierarchisch geordnete, sondern gleichberechtigte Programme, wobei in einem Einprozessorsystem sich jeweils genau eine im Laufzustand befindet. Der fehlende Return-Mechanismus hat zur Folge, daß keine Parameterübergabe wie bei Prozeduren möglich ist. Modula-2 beinhaltet zur Nutzung des Koroutinen-Konzeptes spezielle Sprachelemente in Form von Prozeduraufrufen, die von dem in jedem Modula-2-System vorhandenen Modul SYSTEM exportiert werden /2/. Eine Koroutine wird durch

```
NEWPROCESS (proc          :PROC;
   WorkSpaceAdr           :ADDRESS;
   WorkSpaceSize          :CARDINAL;
   VAR process            :PROCESS)
```

definiert, wobei *proc* eine parameterlose, nicht terminierende Prozedur sein muß; *process* ist der Ausgabeparameter und identifiziert die neue Koroutine. Die explizite Umschaltung zwischen Koroutinen erfolgt mit

```
TRANSFER (VAR process1,
              process2 :PROCESS)
```

Die laufende Koroutine wird unterbrochen, ihr Laufzustand in *process1* abgelegt und *die Koroutine* fortgesetzt, die durch den Wert von *process2* identifiziert wird. Grundsätzlich gilt, daß der Ausführungsteil des Hauptmoduls die Eigenschaft einer Koroutine besitzt (welche als einzige terminieren darf!) und deshalb Ausgangspunkt und Ziel von TRANSFER-Operationen sein kann. Dies ist zum Start der Koroutinen-Arbeit notwendig.

Task-Manipulationen

Das Koroutinen-Konzept ist relativ elementarer Natur und verlangt vom Anwender eine explizite Organisation der Task-Umschaltung sowie der Konstruktion von Synchronisations- und Kommunikationsmechanismen. Wünschenswert ist ein Multitasksystem, das entsprechende Dienste bereitstellt, wobei die Konzeption der Koroutine zur Konstruktion von Tasks benutzt werden kann. Dies soll nachfolgend anhand von Programmfragmenten gezeigt werden. Der Begriff der Task wird dabei als bekannt vorausgesetzt (siehe /3/) und ist u. a. dadurch gekennzeichnet, daß eine Task verschiedene Zustände annehmen kann. Als Teil eines solchen Systems könnte ein Modul TaskHandler geschaffen werden, welcher Definitionen von und Operationen über Tasks beinhalten soll. Der Definitionsteil könnte folgendes Muster besitzen:

```
DEFINITION MODULE TaskHandler;
TYPE TASK;
VAR
      current:TASK;
(* wird durch Hauptmodul initialisiert *)
      size        :CARDINAL;
PROCEDURE CreateTask (proc:PROC;
                      VAR task:TASK);
PROCEDURE Schedule;
(* es folgen gegebenenfalls Importe und
weitere Deklarationen *)
END TaskHandler.
```

Der abstrakte Datentyp TASK repräsentiert im Kern eine Koroutine sowie zusätzliche Datenobjekte zur komfortablen Multitask-Organisation. Die parameterlose Prozedur *proc* wird durch *CreateTask* in eine Task verwandelt und in den Zustand READY überführt. Die Prozedur *Schedule* unterbricht die laufende (RUN-)Task und überführt sie in den Zustand READY. Danach wird die am längsten wartende Task aus dem Zustand READY in den Zustand RUN versetzt. Dies sind natürlich nur stark vereinfachte Konstruktionen, die aber zunächst zur Erläuterung der Problematik ausreichend sind. Ein zugehöriges Programm könnte etwa ein Aussehen wie in Bild 1 haben.

Task-Synchronisation

Jedes Multitasksystem stellt Mechanismen zur Synchronisation von Tasks zur Verfügung. Hier soll am Beispiel des bekannten Semaphores ein Synchronisationsmuster in Modula-2 diskutiert werden. Ziel dieser Konstruktion ist der wechselseitige Ausschluß von Tasks beim Zugriff auf gemeinsame Betriebsmittel.

```
DEFINITION MODULE Semaphor;
TYPE SEMAPHOR;
PROCEDURE Request (VAR s:SEMAPHOR);
PROCEDURE Release (VAR s:SEMAPHOR);
(* eventuell weitere Deklarationen *)
END Semaphor.
```

Mit *Request* wird ein Semaphor besetzt, mit *Release* erfolgt dessen Freigabe. Falls das Semaphor nicht frei ist, wird die anfordernde Task in einer zugehörigen WAIT-Liste verwaltet. Der zugehörige Implementationsmodul hätte dann etwa den Aufbau wie in Bild 2. Auf ähnliche Art können weitere Synchronisations- und Kommunikationsstrukturen aufgebaut werden.

Interrupt-Einbindung

Bei der Entwicklung von Betriebssystemen spielt die Interrupt-Behandlung eine wesent-

```
IMPLEMENTATION MODULE TaskHandler;
FROM ListenHandler IMPORT RUN,READY,FirstTask,GetTask,PutTask;
  (* READY und RUN sind Listen,  GetTask und PutTask tragen Listen-
     elemente ein bzw. aus,  FirstTask kopiert erstes Listenelement *)
FROM SYSTEM  IMPORT TSIZE,ADDRESS,PROCESS,NEWPROCESS,TRANSFER;
FROM Storage IMPORT ALLOCATE;

TYPE TASK = POINTER TO TaskDescriptor;
     TaskDescriptor = RECORD
                        ProcessDescriptor : PROCESS;
                        (* weitere taskspezifische Parameter       *)
                      END (* record *);
VAR old : TASK;

PROCEDURE CreateTask (proc : PROC; VAR task : TASK);
VAR WorkSpaceAdr : ADDRESS;
BEGIN
NEW (task);                        (* Arbeitsspeicherreservierung
                                      für die Taskverwaltung       *)
  WITH task^ DO
    ALLOCATE (WorkSpaceAdr,size); (* Arbeitsspeicherreservierung
                                      für die Task-Routine         *)
    NEWPROCESS (proc,WorkSpaceAdr,size,ProcessDescriptor);
                                   (* ...                          *)
  END (* with *);
  PutTask (READY,task);            (* Task wird in READY-Liste ein-
                                      getragen                     *)
END CreateTask;

PROCEDURE Schedule;
BEGIN
  PutTask (READY,GetTask (RUN));   (* Hole RUN-Task und trage sie
                                      als letzte in READY-Liste ein *)
  PutTask (RUN,GetTask (READY));   (* Austragen  der ersten Task aus
                                      der  READY-Liste  und  in  den
                                      Zustand RUN überführen       *)
  old     := current;              (* alte RUN-Task wird zwischenge-
                                      speichert                    *)
  current := FirstTask (RUN);      (* current  wird gleich der  RUN-
                                      Task gesetzt                 *)
  TRANSFER (old^.ProcessDescriptor,current^.ProcessDescriptor);
END Schedule;

BEGIN
  size := 800H;             (* Ausführungsteil, für Initialisierung! *)
END TaskHandler.
```

Bild 1 Modul TaskHandler

```
IMPLEMENTATION MODULE Semaphor;
FROM ListenHandler IMPORT RUN,READY,STATELIST,
                          FirstTask,GetTask,PutTask,StateListEmpty;
  (* StateListEmpty gibt TRUE zurück, wenn sich kein Listenelement
     in der spezifizierten Liste des Typs STATELIST befindet          *)
FROM SYSTEM IMPORT TRANSFER;

TYPE SEMAPHOR = POINTER TO SemaphorDescriptor;
     SemaphorDescriptor = RECORD
                            Wert         : BOOLEAN;
                            SemaphorWait : STATELIST;
                          END (* record *);
  (* Semaphor besetzt: Wert = TRUE, sonst FALSE;
     SemaphorWait repräsentiert WAIT-Liste des Semaphor               *)

PROCEDURE CreateSem (VAR s : SEMAPHOR);
  (* Einrichten einer Datenstruktur vom Typ SEMAPHOR,  Initialisie-
     ren der Variablen des Descriptors                                *)
END CreateSem;

PROCEDURE Request (VAR s : SEMAPHOR);
BEGIN
  IF s^.Wert THEN               (* wenn Semaphor besetzt ist,         *)
    PutTask (s^.SemaphorWait,GetTask (RUN));
                                (* dann wird die RUN-Task in eine
                                   WAIT-Liste eingetragen             *)
    PutTask (RUN,GetTask (READY));
    old     := current;
    current := FirstTask (RUN);
    TRANSFER (old^.ProcessDescriptor,current^.ProcessDescriptor);
  ELSE
    s^.Wert := TRUE;            (* Besetzen des Semaphor              *)
  END (* if *);
END Request;

PROCEDURE Release (VAR s : SEMAPHOR);
BEGIN
  IF StateListEmpty (s^.SemaphorWait) THEN
                                (* wenn WAIT-Liste leer ist,          *)
    s^.Wert := FALSE;           (* dann Freigabe des Semaphor         *)
  ELSE
    PutTask (READY,GetTask (s^.SemaphorWait));
                                (* ansonsten Task aus WAIT-Liste
                                   holen, und in  die READY-Liste
                                   eintragen, wobei das Semaphor be-
                                   setzt bleibt                       *)
  END (* if *);
END Release;

                                (* Modul Semaphor ohne Ausführungsteil *)
END Semaphor.
```

Bild 2 Modul Semaphor

```
MODULE TestInterrupt;
FROM TaskHandler IMPORT current,size,CreateTask;
FROM ListenHandler IMPORT RUN,READY,
                          FirstTask,PutTask,GetTask;
FROM SYSTEM IMPORT ADDRESS,PROCESS,TRANSFER,IOTRANSFER,NEWPROCESS;
FROM Storage IMPORT ALLOCATE;

CONST IntVector = (* entsprechende Vectoradresse  eines Interrupt  *);
VAR WorkSpace  : ADDRESS;
    t1         : TASK;
    IntHandler : PROCESS;
    main       : PROCESS;
  (* Ausführungsteil eines Hauptmoduls ist Coroutine, deren Identi-
     fikator in main abgelegt wird                                   *)

PROCEDURE task1;
BEGIN
  LOOP
                                     (* ...                          *)
  END;
                                     (* nicht terminierend!          *)
END task1;

PROCEDURE InterruptHandler;
VAR temp : PROCESS;
BEGIN
  temp := main;     (* Rücksprung  zum  Ausführungsteil  des  Haupt-
                       moduls wird vorbereitet                       *)
  LOOP
   IOTRANSFER (IntHandler,temp,IntVector);
                      (* TRANSFER von IntHandler zu temp bei gleich-
                         zeitiger Initialisierung des Interrupt's    *)
   current^.ProcessDescriptor : =temp; (* Retten  des  Descriptors
                                          der unterbrochenen Task    *)
   (* Interruptbehandlung  einschließlich  eventueller Taskmani-
      pulationen                                                     *)
   current := FirstTask(RUN);             (* current wird aktualisiert *)
   temp := current^.ProcessDescriptor; (* Vorbereitung des Rück-
                                          sprungs  aus  der  Inter-
                                          ruptroutine                *)
  END (* loop *);
END InterruptHandler;

BEGIN
  ALLOCATE (WorkSpace,size);  (* Arbeitsspeicherreservierung für
                                 den Interrupt-Handler               *)
  NEWPROCESS (InterruptHandler,WorkSpace,size,IntHandler);
  TRANSFER (main,IntHandler); (* Installierung des Interrupt         *)
  CreateTask (task1,t1);
  (* ...                                                             *)
  PutTask (RUN,GetTask (READY);
  current := FirstTask (RUN);
  TRANSFER (main,current^.ProcessDescriptor);
  (* Start der Multitaskarbeit mit Interrupt                         *)
END TestInterrupt.
```

Bild 3 Modul TestInterrupt

liche Rolle. Sie ist ein wichtiges Kriterium für die Beurteilung von Multitasksystemen. Modula-2 bietet mit der SYSTEM-Prozedur IOTRANSFER die Möglichkeit, die Behandlung von Interrupts auf einem hohen Abstraktionsniveau durchzuführen.

IOTRANSFER (VAR IntHandler:PROCESS; InterruptedProcess:PROCESS; IntVector:CARDINAL)

IOTRANSFER bietet die Möglichkeit, die Auslösung von TRANSFER als Reaktion auf einen Interrupt, spezifiziert durch eine Vektoradresse *IntVector*, zu installieren. Nach dieser Installation wird bei einem Interrupt ein TRANSFER asynchron in die laufende Koroutine zur Koroutine *IntHandler* eingefügt. Der Laufzustand der unterbrochenen Koroutine wird in *InterruptedProcess* abgelegt. Mit dem Aufruf von IOTRANSFER in *IntHandler* kann somit zur vorher unterbrochenen Koroutine zurückgesprungen werden. Im Bild 3 wird ein Hauptmodul vorgestellt, der ein Startpunkt für die Multitaskarbeit ist und gleichzeitig die Möglichkeit der Einbindung von Interrupts in die Multitaskarbeit demonstriert.

Mit dem Übergang zu 16-Bit-Rechnern ist nicht nur eine quantitativ, sondern auch eine qualitativ neue Softwareentwicklung für diese Systeme notwendig. Modula-2 eignet sich besonders für die Entwicklung umfangreicher Systemsoftware einschließlich Echtzeitanwendungen, wobei das Hochsprachenniveau nicht verlassen werden muß. Ein strenges Typkonzept erhöht außerdem die Softwarezuverlässigkeit. Am Wissenschaftsbereich Technische Informatik der Sektion Technische und Biomedizinische Kybernetik der TH Ilmenau wird aufbauend auf den gewonnenen Erfahrungen bei der Entwicklung von multiFORTH /4/ ein konfigurierbares Multitaskbetriebssystem in Modula-2 für die flexibel automatisierte Fertigung entwickelt (vgl. MP 11/89, Seite 342). Ziel ist ein effizientes, prozeßnahes Betriebssystem, wobei größter Wert auf Portabilität und Strukturierung gelegt wird.

Literatur

/1/ Hauptmann, S.: MODULA-2: PASCAL ohne die Nachteile von PASCAL. Mikroprozessortechnik, Berlin 2 (1988) 10, S. 296

/2/ Wirth, N.: Programming in Modula-2. Springer-Verlag, Berlin, Heidelberg, New York 1985

/3/ Werner, D.: Programmierung von Mikrorechnern. Verlag Technik, Berlin, 1983

/4/ Krapp, M.; Richter, J.; Schwartz, J.: Eine FORTH-Systemfamilie. Mikroprozessortechnik, Berlin 2 (1988) 2, S. 53

✉ KONTAKT ☏

Technische Hochschule Ilmenau, Sektion Technische und Biomedizinische Kybernetik, PSF 327, Ilmenau, 6300; Tel. 7 46 47

Kodierung von Wordstar-Files

Rainer Ploss, Berlin

Mitunter besteht die Notwendigkeit der Überführung von Wordstar-Files in andere Formate oder umgekehrt. Abhängig von den speziellen Anforderungen hat dies mit oder ohne Konvertierung der Steuerzeichen zu erfolgen.
Im vorliegenden Beitrag möchte ich auf die Kodierung der Files von Wordstar 3.2x und 3.3x im allgemeinen und die der Steuerzeichen im besonderen sowie auf Probleme beim Erarbeiten von Konvertierungsprogrammen eingehen.

Denkbare Anwendungsbeispiele für solche Konvertierungen sind:

- Übernahme und Übergabe von Daten durch andere Programmsysteme (beispielsweise dBase, Druckvorlagen)
- Überführung von Daten von bzw. auf Betriebssystemebene (beispielsweise zur Weiterverarbeitung unter der Regie von Batch-Files oder zum beschleunigten Drucken von Wordstar-Files)
- Erzeugung von Drucksteuersequenzen ohne die Hilfe von Programmiersprachen
- Korrektur von Programmquelltexten, die fälschlicherweise im Modus **D** statt **N** unter Wordstar bearbeitet wurden.

In einem Wordstar-File sind neben den druckbaren Zeichen (Buchstaben, Ziffern, Sonderzeichen) aus dem 7-Bit-Kode-Zeichensatz, das heißt die Zeichen mit den ASCII-Werten 32 bis 127, auch die Steuerzeichen aus dem 7-Bit-Kode-Zeichensatz kodiert. Es sind dies die Zeichen mit den ASCII-Werten 0–31. Ebenfalls in Wordstar-Files sind weitere Steuerinformationen zur Drucksteuerung und für den Mail-Merge-Druck von Serienbriefen, variablen Texten usw. kodiert, aber auch länderspezifische Zeichen (z. B. Umlaute und ß im deutschen Zeichensatz) aus dem 8-Bit-Kode-Zeichensatz, das heißt die Zeichen mit den ASCII-Werten 128 bis 255. Eine Kennzeichnung für den Wortumbruch und die Neuformatierung von Absätzen, eine Kennzeichnung dafür, daß ein gegebenes Zeichen das Ende eines Wortes darstellt, muß außerdem für jedes dieser Zeichen möglich sein.

Somit reicht ein Byte mit 256 Möglichkeiten für diese Kodierung nicht aus; es wurde deshalb folgendermaßen vorgegangen:
① Kodierung der druckbaren Zeichen und der Steuerzeichen im 7-Bit-Kode
② Nutzung des achten Bits eines jeden Bytes (also auch bei Steuerzeichen!) für die Wortendekennung (gesetzt = Wortende)
③ Ein vorangestelltes Steuerzeichen 27 (1BH) und ein nachgestelltes Steuerzeichen 28 (1CH) kennzeichnet, daß ein Zeichen mit einem Kode größer 127 (7FH) kein Wortende, sondern ein Zeichen aus dem 8-Bit-Kode-Zeichensatz ist.
④ Weitere Steuerinformationen in der Form definierter Sequenzen aus druckbaren Zeichen werden verschlüsselt, indem sie mit einem Punkt in der ersten Position einer Zeile beginnen (Punktbefehle).
Die Steuerzeichen 1–25 (01–19H) entsprechen etwa den Ergebnissen der Sequenzen ^PA bis ^PY. Die konkreten Inhalte werden in der Tafel 1 erläutert.

Die Konvertierung

Bei der Überführung von ASCII-Files nach Wordstar ist nur die Klammerung der Zeichen des 8-Bit-Kode-Zeichensatzes mit 1BH und 1CH zu ergänzen.
Wesentlich komplizierter ist die im weiteren

Tafel 1 Wordstar-File-Kodierung

interner Kode (hexa-dezimal)	Inter-aktion	Bedeutung
01	^PA	Zweite Schreibdichte ein (im allgemeinen Elite)
02	^PB	Schattenschrift (Fettdruck) ein/aus
03	^PC	Druckpause, bis wieder P im Startmenü gedrückt wird
04	^PD	Doppeldruck ein/aus
05	^PE	frei installierbar (maximal 4 Zeichen)
06	^PF	Phantom-Leerzeichen; druckt Zeichen 20H
07	^PG	Phantom-Löschzeichen (Rubout) druckt 7FH
08	^PH	Backspace; überdrucken des nächsten Zeichens
09	^PI	starre Tabulatoren; Sprung auf Position 1,9,17... und Einfügen von Leerzeichen
0A	^PJ	Zeilenvorschub (LF) ohne Wagenrücklauf
0B	^PK	Drucken von Kopf- und Fußzeilen linksbündig bei geradzahligen Seitennummern und rechtsbündig bei ungeradzahligen
0C	^PL	Seitenvorschub (FF)
0D	^PM	Wagenrücklauf (CR) ohne Zeilenvorschub
0E	^PN	Standardschreibdichte ein (im allgemeinen Pica)
0F	^PO	erhält Leerräume beim Neuformatieren eines Absatzes, beispielsweise Karl IV.
10	^PP	Proportionalschrift, nicht dokumentiert
11	^PQ	frei installierbar (maximal 4 Zeichen)
12	^PR	frei installierbar (maximal 4 Zeichen)
13	^PS	Unterstreichen ein/aus
14	^PT	Hochstellen ein/aus
15	^PU	keine Bedeutung
16	^PV	Tiefstellen ein/aus
17	^PW	frei installierbar (maximal 4 Zeichen)
18	^PX	sichtbares Textdurchstreichen mit "–"
19	^PY	Schwarz-Rot-Farbbandumschaltung, wenn vorhanden, sonst frei installierbar
1A	EOF	File-Ende; WordStar verwendet 128-Byte-Sätze, der letzte Satz wird mit EOF-Zeichen vollständig aufgefüllt
1B	ESC	Kennzeichnung, daß das folgende Zeichen aus dem 8-Bit-Zeichensatz ist und sein achtes Bit nicht als Wortende interpretiert wird.
1C	FS	Der Kode 1CH folgt dem Zeichen aus dem 8-Bit-Zeichensatz.
1D	GS	keine Bedeutung
1E	RS	bedingter (weicher) Trennstrich, wenn er innerhalb einer Textzeile steht
1F	US	bedingter (weicher) Trennstrich, wenn er am Ende einer Textzeile steht, das heißt wenn er aktiv ist und ausgedruckt wird

beschriebene *Rückrichtung*. Sie ist abhängig von

- den Anforderungen an die Konvertierung der Steuerzeichen (teilweise, vollständig)
- dem vorliegenden frei installierbaren Teil der Druckersteuerung (^PE, ^PQ, ^PR, ^PW, ^PY)
- dem angeschlossenen Drucker.

Bei dieser Konvertierung werden folgende Gruppierungen erkannt:

- Wortendekennung (Rücksetzen Bit 8)
- Ein-Byte-Eliminierungen (siehe hexadezimaler Kode 1EH)
- Ein-Byte-zu-Ein-Byte-Ersetzungen (siehe 06H, 07H, 08H, 0AH, 0CH, 0DH, 0FH, 1AH und 1FH)
- Ein-Byte-zu-Mehr-Byte-Ersetzungen (siehe 01H und 0EH sowie eventuell 05H, 11H, 12H, 17H, 19H)

– Ein-Byte-zu-Mehr-Byte-Ersetzungen mit Ein-/Ausschaltung (siehe 02H, 04H, 13H und 14H sowie eventuell 05H, 11H, 12H, 17H und 19H)
– Tabulatoren (siehe 09H; Einfügen einer von der aktuellen Position in der Zeile abhängigen Anzahl von 1–8 Leerzeichen)
– Durchstreichen (siehe 18H; n-mal Backspace und Überschreiben mit "–")

- Mehr-Byte-zu-Ein-Byte-Ersetzungen (siehe Rahmen aus 1BH und 1CH)
- Punktbefehle (Interpretation der konkreten Befehle)

Bei der Überführung von Wordstar-Files in ASCII-Files mit teilweiser Konvertierung der Steuerzeichen (reine Textkonvertierung) sind die Wortendekennung, die Punktbefehle, die Klammerung der Zeichen des 8-Bit-Kode-Zeichensatz sowie alle Steuerzeichen von 01H–1FH (außer 0DH, 0AH und 1AH) zu eliminieren bzw. diejenigen Steuerzeichen zu ersetzen, die zu druckbaren Zeichen führen (06H, 07H, 0FH, 1FH sowie Tabulatoren 09H und Durchstreichen 18H). Bei einer vollständigen Konvertierung müssen alle genannten Gruppierungen berücksichtigt werden.

Die meisten Gruppierungen lassen sich ohne programmtechnischen Aufwand lösen, indem die Ausgangsdatei mit Wordstar in eine neue Datei gedruckt wird. Beantwortet man dabei die Frage nach der Wiederverwendung durch Wordstar mit nein, werden alle Wortendekennungen eliminiert, die Punktbefehle entsprechend ihrer Bedeutung aufgelöst und die Kennzeichnungen für die Zeichen des 8-Bit-Kode-Zeichensatz (1BH und 1CH) entfernt. Umgesetzt werden auch die Anweisungen ^PK, ^PI, ^PM und ^PO (sowie ^PL, wenn die Frage nach der Verwendung des Seitenvorschubbefehls verneint wird).

Offen bleibt somit nur noch das Problem der Drucksteuerzeichen 01–07H und 10–19H. Bei der Erarbeitung solcher Konvertierungsprogramme ist sinnvollerweise eine Tabelle analog der Tafel 1. zuzüglich einer Spalte mit den konkreten Sequenzen für den vorliegenden Drucker zu erstellen.

Die konkret installierten Werte der Sequenzen für die Drucksteuerung lassen sich entweder aus den aktuellen Werten des Installationsprogrammes WINSTALL oder durch einen Wordstar-Testlauf und anschließenden Vergleichstestlauf mit dem Druckerhandbuch ermitteln.

Sie können aber auch direkt mit einem Debugger aus dem WS.COM-File entnommen werden /1/, jedoch sind die Positionen abhängig von der vorliegenden Version. (Es gibt beispielsweise Unterschiede zwischen den Versionen 3.32 und 3.24.)

Die Position der Eintragung einer gesuchten Eigenschaft läßt sich durch alternierendes Installieren dieser Eigenschaft und anschließenden File-Vergleich beispielsweise mit dem MS-DOS-Kommando **FC** ermitteln.

Ein in C geschriebenes vorliegendes Konvertierungsprogramm erreicht bei der vollständigen Umwandlung von Files auf 16-Bit-Computern in Abhängigkeit von der Filegröße, der Anzahl der erfolgten Korrekturen, den verwendeten Laufwerken und den Gerätetreibern eine Geschwindigkeit von etwa 1 bis 4 KByte/s auf Disketten, etwa 5 bis 8 KByte/s auf der Festplatte und etwa 11 bis 13 KByte/s auf der RAM-Disk.

Mit Ausnahme der Interpretation der Punktbefehle und des Durchstreichens (^PX) wurden alle Gruppierungen berücksichtigt.

Literatur

/1/ Juergensmeier, G.: Wordstar für PCs und Kompatible, Haar bei München, Markt und Technik Verlag 1986
/2/ MicroPro International Corporation: Wordstar Benutzerhandbuch. Version 3.3, 1982
/3/ MicroPro International Corporation: Wordstar Installationshandbuch. Version 3.3, 1982
/4/ SEIKO EPSON Corporation: Matrixdrucker Handbuch 1987
/5/ Hewlett-Packard-Company: ThinkJet Printernutzerhandbuch, 1984

Neuer Funktionsinterpreter für Turbo-Pascal

Peter E. Nebe, Dresden

Arithmetische Ausdrücke in einen Turbo-Pascal-Quelltext einzuarbeiten, ist kein Problem. Wie lassen sich aber beliebige mathematische Funktionen mit einem kompilierten Programm berechnen? Hier hilft ein eigener Funktionsinterpreter.

Das im folgenden vorgestellte Unterprogramm in Turbo-Pascal analysiert die als Zeichenkette eingegebene Funktion zeichenweise von links nach rechts und berechnet dabei gleichzeitig den Funktionswert. Andere Lösungen, zum Beispiel in /1/, führen Analyse und Berechnung nacheinander durch und erfordern damit einen erhöhten Programmieraufwand.

In der im Bild 1 vorgestellten Funktion wird nach dem Schema Operand-Operator-Operand gesucht und berechnet (Zeilen 159 bis 162), wobei ein Operand eine Zahl, eine Variable x oder y, ein Funktionsaufruf, ein Klammerausdruck oder eine Verkettung von Operanden mit höherer Priorität sein kann (Funktionen *Wert* und *Operand*). Zu diesem Zweck wird vor der eigentlichen Rechenoperation stets getestet, ob der Operator, der dem Schema nachfolgt, höhere Priorität besitzt (Zeile 137).

Der Interpreter arbeitet mit rekursiven Unterprogrammen. Auf den Compiler-Schalter (*$A–*) konnte jedoch verzichtet werden, weil die entsprechenden Variablen dynamisch zwischengespeichert werden (Stapelprinzip, Prozeduren *push* und *pop*). Somit können beliebig komplizierte Ausdrücke berechnet werden. Auch das Potenzzeichen ^ kann benutzt werden (Basis>0). Das Programm läßt sich leicht um weitere Funktionsaufrufe, zum Beispiel zyklometrische und hyperbolische Funktionen, entsprechend TAN und COT erweitern.

Fehler werden zum größten Teil mit der Prozedur *Meldung* abgefangen. Eine Überschreitung des Wertebereiches für Real-Zahlen führt jedoch zum Programmabbruch. Diese Problematik wurde in /2/ behandelt.

```
  1:  Ø FUNCTION Funktionswert (Term: str78; x,y: real): real;
  2:  Ø (*********************************************************)
  3:  Ø (* Funktionsinterpreter   Version 2.0    Von Peter E. Nebe  *)
  4:  Ø (* ====================                 Dresden, Mai 1989  *)
  5:  Ø (* wichtige Bezeichner:                                    *)
  6:  Ø (* Gesamtkette: Wert einer Kette von Operanden, die mit    *)
  7:  Ø (*              Operatoren gleichbleibender oder fallender *)
  8:  Ø (*              Prioritaet verknuepft sind                 *)
  9:  Ø (* Teilkette:   Wert einer Kette von Operanden,            *)
 10:  Ø (*              die mit Operatoren gleicher                *)
 11:  Ø (*              Prioritaet verknuepft sind                 *)
 12:  Ø (* Operand:     Wert eines Operanden, der                  *)
 13:  Ø (*              -eine Zahl,                                *)
 14:  Ø (*              -eine Variable x oder y,                   *)
 15:  Ø (*              -ein Funktionsaufruf,                      *)
 16:  Ø (*              -ein Klammerausdruck oder                  *)
 17:  Ø (*              -eine Teilkette hoeherer Prioritaet        *)
 18:  Ø (*              sein kann                                  *)
 19:  Ø (*********************************************************)
 20:  Ø
 21:  Ø CONST Zahlen:      SET OF char=['Ø'..'9','.'];
 22:  Ø       Funktionen= 'SQR SQRT EXP LN ABS SIN COS TAN COT';
 23:  Ø TYPE  str= STRING[3Ø];
 24:  Ø       link=^stapelelement;
 25:  Ø       stapelelement=RECORD
 26:  1                 re:real; ch:char; by:byte;
 27:  1                 next:link
 28:  1                 END;
 29:  Ø VAR Zwischenwert:               real;
 30:  Ø     zeiger,len,i,c:             integer;
 31:  Ø     Operator,Folgeoperator:     char;
 32:  Ø     Prioritaet,Folgeprioritaet: byte;
 33:  Ø     stapel:                     link;
 34:  Ø
 35:  Ø PROCEDURE push (wert:real; zeichen:char; priori:byte);
 36:  Ø VAR p:link;                  (* Zwischenwert auf Stapel legen *)
 37:  Ø BEGIN new(p);
 38:  1       p^.re:=wert; p^.ch:=zeichen; p^.by:=priori;
 39:  1       p^.next:=stapel;
 40:  1       stapel:=p
 41:  1 END;
 42:  Ø
 43:  Ø PROCEDURE pop(VAR wert:real;VAR zeichen:char;VAR priori:byte);
 44:  Ø VAR p:link;                  (* Zwischenwert vom Stapel holen *)
 45:  Ø BEGIN p:=stapel;
 46:  1       stapel:=p^.next;
 47:  1       wert:=p^.re; zeichen:=p^.ch; priori:=p^.by;
 48:  1       dispose(p)
 49:  1 END;
 50:  Ø
 51:  Ø PROCEDURE Meldung (Meldung:str;p:integer); (* Fehlerroutine *)
 52:  Ø BEGIN IF NOT fehler THEN    (* nur eine Fehlermeldung kommt *)
 53:  1        writeln('Fehler: ',Meldung,': Z=',copy(Term,1,p-1),
 54:  1               un,Term[p],nor,copy(Term,p+1,len-p));
 55:  1       fehler:=true
 56:  1 END;
 57:  Ø

 58:  Ø FUNCTION Gesamtkette:real;                        FORWARD;
 59:  Ø FUNCTION Teilkette (Zwischenwert:real):real; FORWARD;
 60:  Ø
 61:  Ø FUNCTION Zahl:real;                (* liefert Zahlenwert        *)
 62:  Ø VAR z:real;
 63:  Ø BEGIN
 64:  1  i:=zeiger+1;
 65:  1  WHILE Term[i] IN Zahlen DO i:=i+1;
 66:  1  val(copy(Term,zeiger,i-zeiger),z,c);
 67:  1  IF c>Ø THEN Meldung('fehlerhafte Zahl',zeiger);
 68:  1  Zahl:=z;zeiger:=i
 69:  1 END;
 70:  Ø
 71:  Ø FUNCTION Funktionscode:byte;    (* liefert Code eines          *)
 72:  Ø VAR p:byte;                     (* Funktionsaufrufes           *)
 73:  Ø BEGIN
 74:  1  i:=zeiger;
 75:  1  REPEAT i:=i+1 UNTIL (Term[i]='(') OR (i>len);
 76:  1  p:=pos(copy(Term,zeiger,i-zeiger),Funktionen);
 77:  1  IF p=Ø THEN Meldung('ungueltiger Funktionsaufruf',zeiger);
 78:  1  Funktionscode:=p;zeiger:=i+1
 79:  1 END;
 80:  Ø
 81:  Ø FUNCTION Funktion:real;           (* berechnet Funktionsaufruf *)
 82:  Ø VAR a:real;
 83:  Ø BEGIN
 84:  1  CASE Funktionscode OF
 85:  2   1: Funktion:=sqr(Gesamtkette);
 86:  2   5: BEGIN a:=Gesamtkette;IF a>=Ø THEN Funktion:=sqrt(a)
 87:  3        ELSE Meldung('Radikand kleiner als null',zeiger-2) END;
 88:  2   1Ø:Funktion:=exp(Gesamtkette);
 89:  2   14:BEGIN a:=Gesamtkette;IF a>Ø THEN Funktion:=ln(a)
 90:  3        ELSE Meldung('LN(A<=Ø) nicht erlaubt',zeiger-2) END;
 91:  2   17:Funktion:=abs(Gesamtkette);
 92:  2   21:Funktion:=sin(Gesamtkette);
 93:  2   25:Funktion:=cos(Gesamtkette);
 94:  2   29:BEGIN a:=Gesamtkette;
 95:  3       IF cos(a)<>Ø THEN Funktion:=sin(a)/cos(a)      (* TAN *)
 96:  3        ELSE Meldung('TAN((2n+1)*pi/2) nicht erlaubt',zeiger-2)
 97:  3      END;
 98:  2   33:BEGIN a:=Gesamtkette;
 99:  3       IF sin(a)<>Ø THEN Funktion:=cos(a)/sin(a)      (* COT *)
100:  3        ELSE Meldung('COT(n*pi) nicht erlaubt',zeiger-2) END;
101:  2   ELSE Meldung('ungueltiger Funktionsaufruf',zeiger-2)
102:  2  END {case}
103:  1 END; {Funktion}
104:  Ø
105:  Ø FUNCTION Wert:real;             (* liefert Wert fuer Operand *)
106:  Ø BEGIN
107:  1  CASE Term[zeiger] OF
108:  2   'A'..'T':      Wert:=Funktion;
109:  2   'Ø'..'9','.': Wert:=Zahl;
110:  2   'X': BEGIN Wert:=x;zeiger:=zeiger+1 END;
111:  2   'Y': BEGIN Wert:=y;zeiger:=zeiger+1 END;
112:  2   '(': BEGIN zeiger:=zeiger+1;Wert:=Gesamtkette END
113:  2   ELSE Meldung('Operand erwartet',zeiger)
114:  2  END {case}
115:  1 END; {Wert}
116:  Ø
117:  Ø PROCEDURE Symbol (VAR z:char;VAR p:byte); (* ermittelt        *)
118:  Ø BEGIN                          (* Rechenzeichen und Prioritaet *)
119:  1  IF zeiger<len THEN
120:  1   BEGIN z:=Term[zeiger];zeiger:=zeiger+1; (* Operator        *)
121:  2     CASE z OF '+','-': p:=1;   (* Prioritaeten              *)
122:  3               '*','/': p:=2;
123:  3               '^':     p:=3;
124:  3               ')':     p:=Ø    (* Ende Klammerausdruck      *)
125:  3      ELSE BEGIN Meldung('Operator erwartet',zeiger-1);p:=Ø END
126:  3     END {case}
127:  2   END {if then}
128:  1  ELSE p:=Ø                      (* Ende des Terms            *)
129:  1 END;
130:  Ø
131:  Ø FUNCTION Operand:real;          (* liefert Wert              *)
132:  Ø VAR op:real;                    (* eines Operanden           *)
133:  Ø BEGIN
134:  1  IF Term[zeiger]='-' THEN BEGIN zeiger:=zeiger+1;op:=-Wert END
135:  1                        ELSE op:=Wert;
136:  1  Symbol(Folgeoperator,Folgeprioritaet);
137:  1  WHILE Folgeprioritaet>Prioritaet DO op:=Teilkette(op);
138:  1  Operand:=op                    (* Operand kann Teilkette    *)
139:  1 END;                            (* hoeherer Prioritaet sein  *)
140:  Ø
141:  Ø FUNCTION Ergebnis (a,b:real;z:char):real; (* fuehrt Rechen- *)
142:  Ø BEGIN                                     (* operation aus  *)
143:  1  CASE z OF
144:  2   '+':Ergebnis:=a+b;
145:  2   '-':Ergebnis:=a-b;
146:  2   '*':Ergebnis:=a*b;
147:  2   '/':IF b<>Ø THEN Ergebnis:=a/b
148:  2              ELSE Meldung('Division durch null',zeiger-2);
149:  2   '^':IF a>Ø THEN Ergebnis:=exp(ln(a)*b)
150:  2             ELSE Meldung('Basis <=Ø',zeiger-4)
151:  2  END {case}
152:  1 END; {Ergebnis}
153:  Ø
154:  Ø FUNCTION Teilkette,            (* liefert Wert einer Kette    *)
155:  Ø VAR op1,op2:real;              (* gleichbleibender Prioritaet *)
156:  Ø BEGIN
157:  1  push(op1,Operator,Prioritaet);(* ersten Operanden,Operator *)
158:  1  REPEAT                         (* und Prioritaet merken      *)
159:  2   op1:=Zwischenwert;
160:  2   Operator:=Folgeoperator;Prioritaet:=Folgeprioritaet;
161:  2   op2:=Operand;
162:  2   Zwischenwert:=Ergebnis(op1,op2,Operator)
163:  2  UNTIL Folgeprioritaet<Prioritaet; (* bis Ende Teilkette     *)
164:  1  Teilkette:=Zwischenwert;
165:  1  pop(op1,Operator,Prioritaet)  (* alten ersten Operanden     *)
166:  1 END;                            (* usw. wiederherstellen      *)
167:  Ø
168:  Ø FUNCTION Gesamtkette;           (* liefert Wert einer Kette  *)
169:  Ø BEGIN                           (* gleichbleibender oder     *)
170:  1  Zwischenwert:=Operand;         (* fallender Prioritaet bis  *)
171:  1  WHILE Folgeprioritaet>Ø DO     (* Ende Klammer oder Term    *)
172:  1   Zwischenwert:=Teilkette(Zwischenwert);
173:  1  Gesamtkette:=Zwischenwert
174:  1 END;
175:  Ø
176:  Ø BEGIN (* Funktionswert *)
177:  1  zeiger:=1;len:=length(Term);stapel:=NIL;fehler:=false;
178:  1  Funktionswert:=Gesamtkette
179:  1 END;  (* Funktionswert *)
```

Bild 1 Der Funktionsinterpreter

```
  1:  Ø PROGRAM Berechnung;   (* Zur Demonstration von FUNKTION.INC *)
  2:  Ø CONST un{terstrichen}= #27'[4m';  nor{mal}= #27'[Øm';
  3:  Ø                                   (* Steuerzeichen fuer A 71ØØ *)
  4:  Ø TYPE  str78=  STRING[78];
  5:  Ø VAR   Formel: str78;
  6:  Ø       x,y,z:  real;
  7:  Ø       i:      integer;
  8:  Ø       fehler: boolean;
  9:  Ø
 1Ø:  Ø {$I FUNKTION.INC}
 11:  Ø
 12:  Ø BEGIN
 13:  1  write('Z='); readln(Formel);
 14:  1  FOR i:=1 TO length(Formel) DO  (* umwandeln in              *)
 15:  1   Formel[i]:=upcase(Formel[i]); (* Grossbuchstaben           *)
 16:  1  write('X='); read(x); write('    Y='); readln(y);
 17:  1  z:=Funktionswert(Formel,x,y);
 18:  1  IF NOT fehler THEN writeln('Z=',z)
 19:  1 END.
```

Bild 2 Demonstrationsprogramm

Literatur

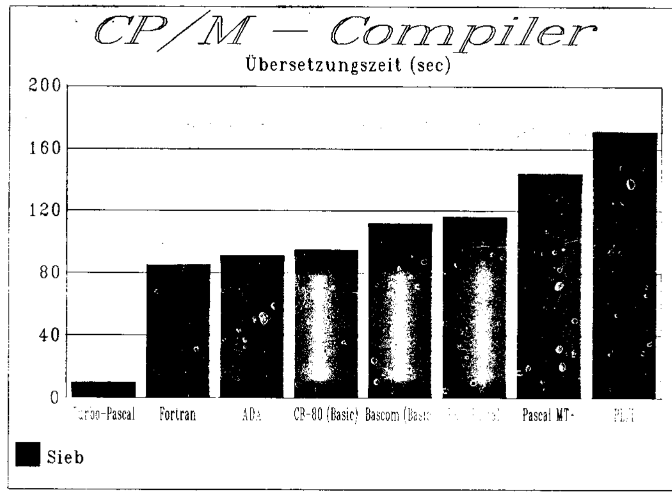
/1/ Röhner, G.: Funktionsinterpreter in Turbo-Pascal. mc, München (1987) 5, S. 78

/2/ Weber, K.-D.: Nutzereigene Laufzeitfehlerbehandlung bei mathematischer Software. Mikroprozessortechnik, Berlin 3 (1989) 4, S. 105

Im Test:

Programmiersprachen für CP/M

Uwe Schulze, Berlin



Nach der Vorstellung der C-Compiler in MP 1/90, Seite 9, soll nun ein Blick auf die anderen Sprachen unter CP/M geworfen werden. Als Test werden wieder die Primzahlen zwischen 3 und 16381 nach Eratosthenes in 10 Iterationen ermittelt, die Fibonacci-Funktion bis 22 protokolliert und 32 000 Bildschirmzeichen ausgegeben (siehe Bilder 1 bis 4). Die Angaben in den Tafeln 1 bis 3 beziehen sich auf eine 2,5-MHz-CPU und das Betriebssystem CP/A mit 2 KByte Blockung.
In Fortran ist die Rekursion nicht definiert, und in Pascal MT+ liefert sie keine korrekten Ergebnisse. Auch die meisten Basic-Implementationen unterstützen sie nicht. Für fig-Forth ist ein Feld der Dimension 8190 auch mit einem System-Subset zu groß.

Außer Turbo-Pascal und Ada erzeugen alle Compiler Microsoftkompatiblen REL-Code; der Weg über Assembler-Zwischencode – wie bei den C-Compilern üblich – wird von

```
procedure prime_factors is
k : integer;
prime : integer;
count : integer;
size : constant integer := 8190;
maxi : constant integer := 127;
flags : array (0..size) of boolean;
c : character;

for k use at 16#80#;
for prime use at 16#82#;
for count use at 16#84#;
for flags use at 16#6000#;

begin
   put("10 iterations");new_line;
   get(c);
   for iter in 1..10 loop
       count := 0;
       for i in 0..size loop
           flags(i) := true;
       end loop;
       for i in 0..size loop
           if flags(i) then
              prime := i + i + 3;
              count := count + 1;
              k := i + prime;
              while k <= size loop
                    flags(k) := false;
                    k := k + prime;
              end loop;
           end if;
       end loop;
   end loop;
put(count);
put(" primes");
new_line;

end prime_factors;
```

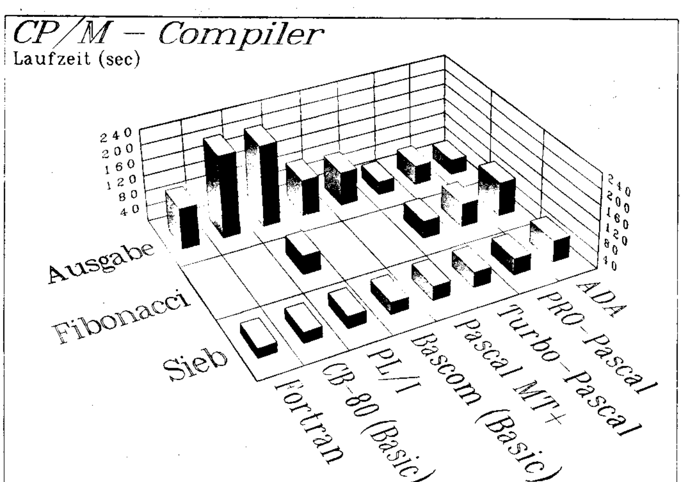
***Bild 1** Primzahlensieb in Ada: gut strukturiert*

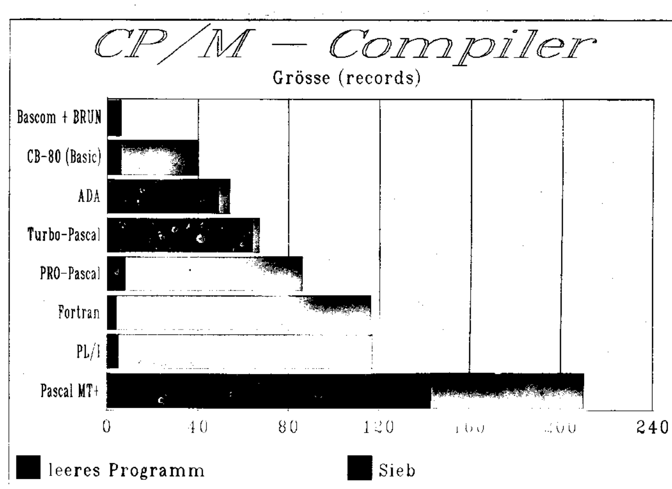
Tafel 1 CP/M-80-Compiler, geordnet nach der Laufzeit des Primzahlensiebes

Compiler	Primzahlensieb Laufzeit in min	Compilezeit in min	Codegröße	Fibonacci-funktion Laufzeit in min	Bildschirm-ausgabe Laufzeit in min	Codegröße leeres Programm
Fortran	0:23	1:25	16 KByte (116 records)	---	1:47	2 KByte (4 records)
CB-80 (Basic)	0:26	1:35	6 KByte (40 records)	---	3:34	2 KByte (6 records)
PL/I	0:30	2:51	16 KByte (117 records)	0:47	3:32	2 KByte (5 records)
BASCOM (Basic)	0:30	1:52	2 KByte + BRUN (6 records)	---	1:36	2 KByte (4 records)
Pascal MT+	0:37	2:24	28 KByte (210 records)	---	1:28	18 KByte (143 records)
Turbo-Pascal	0:39	0:01 MEM 0:10 COM	10 KByte (67 records)	0:45	0:35	8 KByte (64 records)
PRO-Pascal	0:41	1:56	12 KByte (86 records)	1:02	0:51	2 KByte (8 records)
Ada	0:56	1:31	8 KByte (54 records)	1:29	0:39	8 KByte (49 records)

Tafel 2 Zum Vergleich: Z 80-Assembler

Primzahlensieb Laufzeit in min	Compilezeit in min	Codegröße	Fibonaccifunktion Laufzeit in min	Bildschirm-ausgabe Laufzeit in min	Codegröße leeres Programm
0:07	0:46	2KByte (2 records)	0:10	über BDOS: 0:35 über BIOS: 0:17 direkt: 0:03	0 KByte





```
5   DEFINT A-Z
10  dim flags(8190)
20  print "10 iterations"
30  input c
40  for iter=1 to 10
50      count=0
60      for i=0 to 8190
70          flags(i)=1
80      next i
90      for i=0 to 8190
100         if flags(i)=0 then goto 180
110            prime=i+i+3
120            k=i+prime
130            while k<= 8190
140                  flags(k)=0
150                  k=k+prime
160            wend
170            count=count+1
180     next i
190 next iter
200 print count," primes"
210 end
```

Bild 2 Primzahlensieb in Basic: erstaunlich schnell

```
8190 CONSTANT SIZE
0 VARIABLE COUNTER
0 VARIABLE FLAGS   SIZE ALLOT

: ONE-SIEVE
  FLAGS SIZE 1 FILL
  0 SIZE 0
  DO FLAGS I + C@
     IF I DUP + 3 + DUP I +
        BEGIN DUP SIZE <
        WHILE 0 OVER FLAGS + C! OVER + REPEAT
        DROP DROP 1 +
     ENDIF
  LOOP
  COUNTER ! ;

: SIEVE 10 0 DO ONE-SIEVE LOOP COUNTER ? ." PRIMES" ;
```

Bild 3 Primzahlensieb in Forth

```
(DEFUN FIB (LAMBDA(X)
  ((GREATERP 3 X) 1)
   (PLUS (FIB (DIFFERENCE X 1))
         (FIB (DIFFERENCE X 2))) ) )

(SETQ I 0)

(LOOP (PRINT (FIB (SETQ I (PLUS I 1))))
      (UNTIL (EQ I 22) NIL ))
```

Bild 4 Protokollierung der Fibonacci-Funktion in Lisp: kurz, aber langsam

Tafel 3 Interpreter-Sprachen

Sprache	Primzahlensieb Laufzeit in min	Fibonacci-Funktion Laufzeit in min
MBASIC	45:20	---
comForth	2:15	0:50
figForth	---	0:56
Z80-Forth by Ray Duncan	1:57	0:48
MuLISP	---	6:12

keinem genutzt. Auffallend ist, daß Fortran und PRO-Pascal die Datenbereiche im COM-File reservieren: eine raumgreifende Methode. So reduziert sich in Fortran das COM-File ohne das 8190 Byte große Feld um 8 KByte auf die Hälfte der Größe.
Leider standen mir keine Compiler für Algol 68 (früher bevorzugte Lehrsprache) und Modula (der Inbegriff strukturierter Programmierung – z. B. Turbo-Modula) zur Verfügung.

Die serielle Schnittstelle unter DOS

Asynchrone Datenübertragung mit IBM-kompatiblen PCs

Ralf Bergmann, Mathias Weidner, Leipzig

Vom Hersteller werden die meisten der IBM-kompatiblen PCs mit einer oder mehreren Schnittstellen für asynchrone Datenübertragung (COMn) ausgerüstet. Den Gebrauch dieser Schnittstellen, insbesondere der RS-232C-(V.24-)Schnittstelle, etwas näher zu erläutern soll Aufgabe dieses Beitrages sein.

Ralf Bergmann – Jahrgang 1960 – studierte von 1982 bis 1986 an der Technischen Hochschule Leipzig in der Fachrichtung Technische Kybernetik/Automatisierungstechnik. Seit 1986 ist er an der Sektion Naturwissenschaften dieser Hochschule als befristeter Assistent auf dem Gebiet der Laborautomatisierung tätig.

Mathias Weidner – Jahrgang 1962 – studierte von 1985 bis 1989 ebenfalls an der Technischen Hochschule Leipzig in der Fachrichtung Technische Kybernetik/Automatisierungstechnik. Zur Zeit ist er Forschungsstudent an der Sektion Naturwissenschaften der Technischen Hochschule Leipzig.

Die RS-232C-Verbindung

Die Verbindung ist in Europa als V.24 bekannt und in der DDR in /1/ fixiert. Nach diesem Standard werden die Daten seriell, das heißt Bit für Bit, das niederwertigste Bit zuerst, auf einer Leitung übertragen. Eine logische 1 wird durch einen Spannungspegel von weniger als −3V und logisch 0 durch mehr als +3V repräsentiert. Alle Leitungen der Schnittstellen sind lowaktiv. Der Standardsteckverbinder für die RS-232C-Schnittstellen ist der 25polige DB-25S-Steckverbinder.
In Tafel 1 sind die gebräuchlichsten Leitungen mit Pinbelegung und Bezeichnung der Leitung nach DIN (66020), CCITT (V.24) und EIA (RS-232C) aufgeführt. Weiterhin ist vermerkt, ob es sich bei einem Terminalanschluß um einen Eingang (in) oder einen Ausgang (out) handelt. Da die RS-232C-Verbindung ursprünglich zum Anschluß eines Terminals an ein Modem zur Datenfernübertragung vorgesehen war, unterscheidet man bei der RS-232C-Schnittstelle den Terminalanschluß vom Modemanschluß, die miteinander durch ein Eins-zu-eins-Kabel verbunden werden.

Tafel 1 Belegung des RS-232C-Steckverbinders

Pin	Bezeichnung		DIN 66020	CCITT V.24	EIA RS-232C	Terminal
1	Schutzerde	PG	E1	101	AA	
2	Sendedaten	T×D	D1	103	BA	out
3	Empfangsdaten	R×D	D2	104	BB	in
4	Sender einschalten	RTS	S2	105	CA	out
5	Sendebereitschaft	CTS	M2	106	CB	in
6	Betriebsbereitschaft	DSR	M1	107	CC	in
7	Betriebserde	SG	E2	102	AB	
8	Signalträger erkannt	DCD	M5	109	CF	in
20	Empfänger bereit	DTR	S1	108	CD	out
22	ankommender Ruf	RI	M3	125	DE	in

DIN – Deutsches Institut für Normung (BRD)
CCITT – Comité Consultatif International Télégraphique et Téléphonique (Frankreich)
EIA – Electronic Industries Association

Die Hardwareablaufsteuerung arbeitet im Fall einer Terminal-Modem-Verbindung folgendermaßen: Wenn das Terminal senden möchte, aktiviert es Pin 2 (RTS). Sobald das Modem empfangen kann, aktiviert es Pin 5 (CTS). Das Terminal sendet nur, wenn CTS aktiviert ist. Wenn das Modem senden möchte, aktiviert es Pin 8 (DCD). Ist das Terminal empfangsbereit, so aktiviert es Pin 20 (DTR). Das Modem sendet nur, wenn DTR aktiviert ist. Über Pin 22 (RI) wird dem Terminal mitgeteilt, daß ein Anruf von der externen DFÜ-Verbindung des Modems kommt. Pin 6 (DSR) wird vom Modem zugleich mit DCD aktiviert.
Eigentlich sollte man einen Terminalanschluß von einem Modemanschluß bereits am Stecker unterscheiden können. Normalerweise sind am Terminal Stecker und am Modem Buchse angebracht. Leider halten sich nicht alle Hersteller an diese Konvention, im Zweifelsfalle sollte man Pin 2 (TxD) untersuchen. Beim Terminal ist dieser Anschluß ein Leitungssender, beim Modem ein Empfänger. Der Pegel an einem Sendepin beträgt im Ruhezustand −12V. Die Spannung an einem unbeschalteten Empfangspin liegt im Bereich zwischen −3V und +3V.
Da die RS-232-C-Verbindung auch geeignet ist, zwei Computer direkt miteinander zu verbinden, ohne Modems zwischenzuschalten, wurde das Null-Modem-Kabel (NMC) eingeführt, bei dem auf beiden Seiten Terminals angeschlossen werden. Für dieses Kabel gibt es verschiedene Varianten, je nach dem,

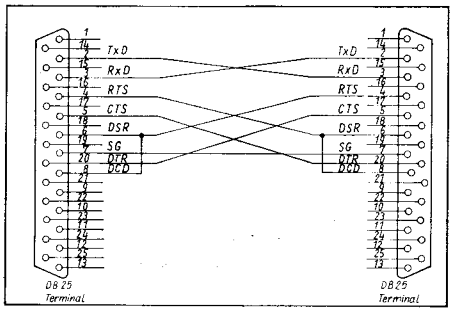


Bild 1 Full-Null-Modem-Kabel

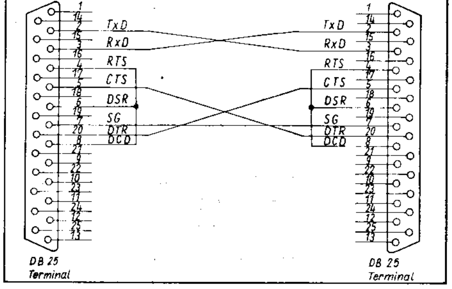


Bild 2 Empfohlenes Null-Modem-Kabel

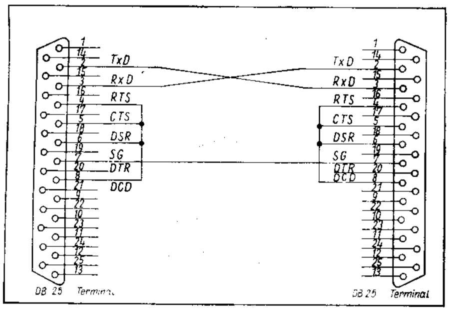


Bild 3 Empfohlenes Terminalkabel (es kann ständig gesendet werden)

wie weit die Hardwareablaufsteuerung ausgebaut werden soll. Generelles Kennzeichen eines Null-Modem-Kabels sind die gekreuzten Leitungen RxD und TxD, die Pins 7 (SG) werden immer miteinander verbunden (Signalrückleiter).
Beim Full-NMC werden zusätzlich die Pins 4 (RTS) mit 8 (DCD) und Pin 20 (DTR) mit Pin 5 (CTS) über Kreuz verbunden. Um auf der sicheren Seite zu sein, werden auf jeder Seite Pin 6 (DSR) mit Pin 8 verbunden (Bild 1). Wenn beide Teilnehmer ständig empfangsbereit sind, kann man das DCD- (und das DSR-) Signal ständig generieren, indem man diese Leitungen mit dem RTS-Signal des eigenen Anschlusses verbindet (Bild 2). DTR und CTS dienen in diesem Fall als *Busy*-Signale. Die einfachste Verbindung enthält nur die Leitungen RxD und TxD (Bild 3). Dieses Kabel kommt mit 3 Adern aus.

Der Schnittstellenbaustein des PCs

In 16-Bit-PCs wird für die serielle Schnittstelle üblicherweise der universelle asynchrone Sende- und Empfangsschaltkreis (universal acynchronous receiver transmitter; UART) 8250 eingesetzt. Hier werden Zeichen, die gesendet werden sollen, zunächst in ein Halteregister (transfer holding register; THR) eingeschrieben. Aus dem Halteregister werden sie in ein Schieberegister (transfer shift register; TSR) übernommen, von dem sie Bit für Bit auf die Leitung gelegt werden. Vor jedem Zeichen wird ein Startbit (Low) eingefügt. Nach jedem Zeichen folgen 1, 1,5 oder 2 Stoppbits (High). Das niederwertigste Bit (LSB) des Zeichens wird als erstes und das höchstwertige Bit (MSB) wird zuletzt gesendet (Bild 4). Ein neues Zeichen wird erst dann in das TSR übernommen, wenn das vorhergehende vollständig übertragen wurde.
Wenn an der Empfangsleitung ein Low-Signal (Startbit) auftritt, beginnt der UART die folgenden Daten Bit für Bit in ein Empfangsschieberegister (receiver shift register; RSR) einzuschreiben. Danach wird das empfangene Zeichen in ein Empfangsdatenregister (receiver data register; RDR) übernommen. Wenn am Ende eines Zeichens kein Stoppbit erkannt wurde, so wird ein Protokollfehler (framing error) angezeigt. Wenn ein Zeichen in das RDR übernommen wird, bevor das vorhergehende Zeichen ausgelesen wurde, wird ein Überlauffehler (overflow) angezeigt.
Zeichen können mit gerader, ungerader oder ohne Parität gesendet und empfangen werden. Dabei wird das letzte vor den Stoppbits gesendete Bit beim Senden so gesetzt bzw. rückgesetzt, daß die Anzahl der High-Bits der gewählten Parität entspricht. Falls beim Empfang dieses Bit nicht der gewählten Parität entspricht, wird ein Paritätsfehler angezeigt.
Für die seriellen Schnittstellen existieren im Hauptspeicher Zeiger, die die Ein-/Ausgabe-Adresse der betreffenden Schnittstelle beinhalten. Dabei gilt folgende Zuordnung:

0040:0000	**:E/A-Adresse der 1. RS-232C-Schnittstelle**
0040:0002	**:E/A-Adresse der 2. RS-232C-Schnittstelle**
0040:0004	**:E/A-Adresse der 3. RS-232C-Schnittstelle**
0040:0006	**:E/A-Adresse der 4. RS-232C-Schnittstelle**

Das E/A-Tor, auf das der jeweilige Zeiger weist, dient beim Senden als THR und beim Empfangen als RDR. Das Leitungsstatusbyte (Tafel 2) ist an der Schnittstellenadresse zuzüglich eines Offsets von 5 auslesbar. Für den Modemstatus (Tafel 3) beträgt der Offset 6.

Bedienung der COM-Schnittstelle mit DOS-Funktionen

Die COM1-Schittstelle stellt eine serielle Schnittstelle nach RS-232C dar. Vom Betriebssystem MS-DOS wird diese Schnittstelle jedoch nicht sehr komfortabel unterstützt. Bei den DOS-Funktionen, die über den Interrupt 21H aufgerufen werden, existieren nur zwei ungepufferte Hilfsfunktionen für das Lesen und das Schreiben zum Hilfsport AUX, der standardmäßig COM1 zugewiesen ist. Mit dem MODE-Befehl ist eine Umlenkung auf andere Geräte möglich.
Mit der DOS-Funktion 3 wird ein Zeichen vom Hilfsport AUX gelesen, es erfolgt keine Statusmeldung. Beim Aufruf enthält Register AH den Wert 03H, als Ergebnis enthält Register AL das empfangene Zeichen. Die Routine wartet, bis ein Zeichen angekommen ist.
Bei der DOS-Funktion 4 wird ein Zeichen zum Hilfsport AUX geschrieben. Beim Aufruf enthält Register AH den Wert 04H und Register DL das zu sendende Zeichen. Das Zeichen wird ohne Rückmeldung der Funktion gesendet.
Ein Abbruch der DOS-Funktionen 3 und 4 ist mit Ctrl-C möglich. Bei beiden Funktionen ist keine Änderung der Übertragungsparameter möglich, dazu kann der MODE-Befehl genutzt werden. Die Syntax dazu ist:

MODE port:[bitrate],[parität],[datenbits],[stoppbits] mit folgenden möglichen Werten
port :**COM1,COM2**
bitrate:**110,150,300,600,1200,2400,4800,9600**
parität : **e**(ven),**o**(dd),**n**(one)
datenbits : **7,8**
stoppbits : **1,2**

Durch das Betriebssystem werden meistens 9600 Baud, keine Parität, 1 Stoppbit, 8 Datenbits voreingestellt.

... und mit BIOS-Funktionen

Ein paar Möglichkeiten mehr bietet der BIOS-Interrupt 14H. Mit diesem Interrupt ist es möglich, aus einem Programm heraus die oben genannten Parameter der Schnittstelle einzustellen, Zeichen zu senden und zu empfangen sowie den Schnittstellenstatus abzufragen.
Beim Aufruf des Schnittstelleninterrupts 14H wird in Register AH die Funktionsnummer und in Register DX immer die Portnummer übergeben. Dabei entspricht COM1 = 0, COM2 = 1. Von der Funktion wird in Register AH immer der Leitungsstatus zurückgegeben, der in Tafel 2 näher erläutert wird. Bei allen Funktionen bleibt der Inhalt der Register BX, CX, DX, SI, DI, BP und der Segmentregister erhalten.
Mit der Funktion 0 kann die Schnittstelle eingestellt werden. Dabei wird beim Aufruf in Register AH der Wert 00H und in Register AL ein Parameterbyte übergeben, das in Tafel 4 näher erläutert wird. In Register AH wird der Leitungsstatus (vgl. Tafel 2) und in Register AL der Modemstatus (vgl. Tafel 3) zurückgegeben. Beim Aufruf dieser Funktion wird RTS aktiviert und DTR rückgesetzt.
Mit der Funktion 1 kann ein Byte gesendet werden. Dabei wird beim Aufruf des Interrupts in Register 01H und in Register AL das zu sendende Zeichen übergeben. Als Ergebnis wird in Register AH der Leitungsstatus zurückgegeben. Wenn das Zeichen nicht gesendet werden konnte, ist Bit 7 gesetzt (Time-out-Fehler). Beim Aufruf dieser Funktion wird RTS und eventuell auch DTR aktiviert. Das Zeichen wird nur gesendet, wenn CTS aktiviert ist und das vorhergehende Zeichen bereits vollständig gesendet wurde (Transfer-Shift leer).
Mit der Funktion 2 kann ein Zeichen empfangen werden. Als Ergebnis des Funktionsaufrufes erhält man in Register AH den Leitungsstatus und in AL das empfangene Zeichen. Wenn innerhalb der Time-out-Zeit kein Zeichen empfangen wurde, so ist in Register AH Bit 7 gesetzt. Das Zeichen wird erst gelesen,

Tafel 2 Leitungsstatus bei COM-Funktionen

Bit	Bedeutung
7	Time-out-Fehler
6	Transfer-Shift-Register leer
5	Transfer-Hold-Register leer
4	Unterbrechungsfehler
3	Framingfehler (Protokoll nicht eingehalten)
2	Paritätsfehler
1	Überlauf
0	Daten bereit

Tafel 3 Modemstatus bei COM-Funktionen

Bit	Bedeutung
7	RLSD (receive line sign) entspricht DCD Verbindung zum Empfänger-Modem aufgebaut
6	RI (ring indicator) ankommender Ruf Telefon läutet
5	DSR (data set ready) Modem ist angeschaltet
4	CTS (clear to send) Modem ist zum Senden bereit
3	DRLSD (delta RLSD)
2	TERI (trailing edge ring indicator)
1	DDSR (delta DSR)
0	DCTS (delta CTS)

Bits 0 bis 3 sind gesetzt, wenn das jeweilige korrespondierende Bit 4 bis 7 sich seit dem letzten Auslesen des Modemstatus geändert hat.

Tafel 4 Parameterbyte für die Schnittstelleninitialisierung

Bit	Bedeutung		
7..5	Bitrate:	000	– 110 Baud
		001	– 150 Baud
		010	– 300 Baud
		011	– 600 Baud
		100	– 1200 Baud
		101	– 2400 Baud
		110	– 4800 Baud
		111	– 9600 Baud
4	Parität:	0	– ungerade (odd)
		1	– gerade (even)
3		0	– keine Parität (none)
		1	– Parität entsprechend Bit 4
2	Stoppbits:	0	– 1 Stoppbit
		1	– 2 Stoppbits
1,0	Datenbits:	10	– 7 Datenbits
		11	– 8 Datenbits

```
type
 RegType     = record    (* Parameter fuer Intr - Prozedur *)
                case Byte of
                 1:(AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags:Integer);
                 2:(AL,AH,BL,BH,CL,CH,DL,DH:Byte);
                end;
const
 DrvInt      = $14;(* Nummer des Einheitentreiber - Interrupts *)

var
 Regs          : RegType;
                          (* Parameter fuer Intr() - Prozedur *)
 LStatus       : Byte;

Procedure ComPutCh(PNum:Byte;
                         PCh:Char;
                var Leitungsstatus:Byte);
                     (* Zeichen senden *)
begin
 with Regs do begin
  DX:=PNum;
  AL:=ord(PCh);
  AH:=$01;Intr(DrvInt,Regs);
  LeitungsStatus:=AH;
 end
end;  (* PutCh *)

(* Aufruf *)
 ...
ComPutCh(0,'@',LStatus);  (* Portnummer 0 für COM1 *)
 ...
```

Bild 4 Aufruf des Interrupts 14H in Turbo-Pascal

```
#include <bios.h>

int comwert;
char lstatus;
 ...
comwert = bioscom(1,'@',0);
lstatus = (comwert >> 8) & 0xff;
                              /* Leitungsstatus = Bit 8 - 15 */
comwert &= 0xff;              /* Bit 8 - 15 ausblenden */
 ...
```

Bild 5 Aufruf des Interrupts 14H in Turbo-C

wenn sowohl Bit 0 des Leitungsstatusregisters (Daten bereit) als auch Bit 5 des Modem-Statusregisters (DSR) gesetzt sind. Beim Aufruf der Funktion wird DTR aktiviert, falls kein Zeichen zur Verfügung steht. Wenn ein Zeichen gelesen wurde oder nach einem Time-out wird DTR rückgesetzt.
Durch die Funktion 3 kann der Modemstatus abgefragt werden. Als Ergebnis erhält man in Register AL den Modemstatus und in AH den Leitungsstatus (vgl. Tafeln 2 und 3).
Die Wartezeit (Time-out-Zeit) wird für COM1 in RAM-Zelle 0000:047CH und für COM2 in RAM-Zelle 0000:047DH spezifiziert und kann verändert werden. Die Wartezeit ergibt sich aus dem Wert in der betreffenden Speicherzelle mit etwa 0,5 Sekunden multipliziert.

Der Aufruf des BIOS-Interrupts 14H

Nachfolgend soll an einem Beispiel erläutert werden, wie die COM-Schnittstellen aus Turbo-Pascal (Version 3.0), aus Turbo-C und aus MASM heraus aufgerufen werden. Im Beispiel (siehe Bild 4) wird vorausgesetzt, daß die COM-Schnittstelle bereits initialisiert wurde, es soll das ASCII-Zeichen @ (040H) gesendet werden. Um die COM-Schnittstelle zu initialisieren bzw. ein Zeichen einzulesen oder den Status zu ermitteln, muß lediglich beim Aufruf der Schnittstellenfunktion in Register AH der entsprechende Wert übergeben werden (wie bereits oben erläutert).
In Turbo-Pascal wird eine Variable Regs vom Typ RegType (s. u.) vereinbart, mit deren Hilfe der Prozedur Intr (Aufruf eines BIOS-Interrupts) die Registerwerte übergeben werden. Der Prozedur Intr werden als Parameter die Nummer des Interrupts und mit Regs die Registerwerte übergeben. Die Schnittstellenfunktion wird in die Prozedur ComPutCH() eingekleidet. Diese bekommt als Parameter die Portnummer, das zu sendende Zeichen und eine Variable vom Typ Byte, in der der Leitungsstatus zurückgegeben wird.
Bei Turbo-C existiert eine Funktion bioscom() in der Standardbibliothek, die im File bios.h definiert ist als

```
int bioscom (cmd, abyte, port)
       int cmd, port;
       char abyte;
```

Dabei bedeutet cmd die Nummer der Funktion (0: Initialisierung, 1: Senden, 2: Empfangen, 3: Statusabfrage) entsprechend den Werten für Register AH, *abyte* ist das Zeichen, das gesendet werden soll bzw. das Parameterbyte für die Initialisierung, und *port* ist die Portnummer (0 für COM1 ...). Diese Funktion gibt einen 16-Bit-Integerwert zurück, bei dem die oberen 8 Bit die Statusbits sind, während die unteren 8 bit von *cmd* abhängen (z. B. empfangenes Zeichen bei cmd = 2). In diesem Fall würde der Aufruf zu unserem Beispiel aussehen, wie in Bild 5 dargestellt.
Der zurückgegebene Wert muß zur Weiterverarbeitung umgewandelt werden.
Am einfachsten ist der Aufruf eines BIOS-Interrupts aus einem Assemblerprogramm heraus. Für das Beispiel wäre die entsprechende Befehlsfolge

```
...
mov ah,1
mov al,'@'
mov dx,0
int 14H
...
```

Für die anderen Schnittstellenfunktionen des Interrupts 14H sind diese Beispiele in geeigneter Form zu modifizieren.

Literatur

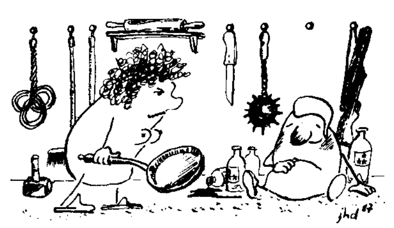
/1/ TGL 29077: Elektrische Informationstechnik, Schnittstelle zwischen Datenendeinrichtung und Datenübertragungseinrichtung. November 1975
/2/ Bogatz, A.: Mikrorechner in der Amateurmeßtechnik. Berlin: Militärverlag 1988
/3/ Hübener, J.: MS-DOS. Berlin: Verlag Technik 1988
/4/ Zietlow, M.: Am Draht zur Welt: Schnittstellen-Interrupt 14thex. DOS EXTRA, München 2 (1988) 5, S. 44
/5/ Hepp, M.: Die serielle PC-Schnittstelle. DOS EXTRA, München 3 (1989) 7, S. 48
/6/ Hartwig, O.: PC/XT/AT für Insider. Haar: Markt & Technik Verlag 1987

✉ KONTAKT ☏

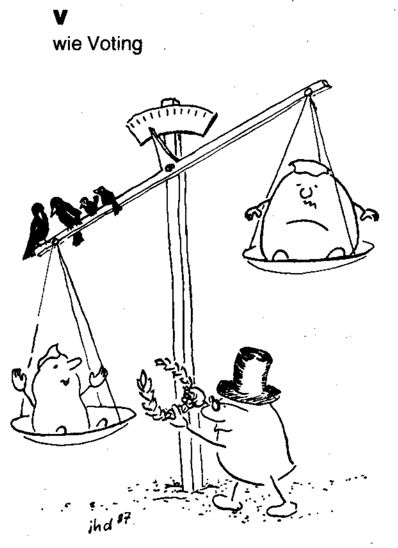
Technische Hochschule Leipzig, Sektion Naturwissenschaften/Zentraler wissenschaftlicher Gerätebau, Karl-Liebknecht-Straße 132, Leipzig, 7030; Tel. 5 62 92 03

Kleines Lexikon der Mikrorechentechnik

U
wie Utilities



V
wie Voting



Zeichnungen: Dahmen

Systeme mit System

Teil 1

Personalcomputer von Low-Cost bis High-Tech

Detlef Grell

Das Motto der folgenden Beiträge lautet etwa „Wie finde ich den für mich passenden PC oder AT mit der richtigen Ausstattung?" Zur Einstimmung ein richtig „böses" Negativbeispiel.

Irrwege

Die Vielfalt erhältlicher PC-Komponenten, die nicht unmittelbar zur Basisplatine im PC oder AT gehören, läßt mannigfaltige Kombinationen zu. Um superpreiswerte 80286-ATs anbieten zu können, kann man diese absolut minimal ausstatten: 512 KByte langsame RAMs, so daß ein Wait-State nötig wird, und nur ein Floppy-Laufwerk für XTs mit entsprechendem Controller.

Selbst ein langjähriger PC-Anwender muß nicht gleich durchschauen können, was er sich da Merkwürdiges nach Hause geschleppt hat: „Wait-States" sagt ihm nichts (gibt's beim PC nicht), und was ist an einem Laufwerk für 360-KByte-Disketten verdächtig? Hat er vorher auch gehabt. Und die Festplatte samt Controller wird nachgerüstet, sobald das Budget es erlaubt.

Der Käufer hat zuerst noch 512 KByte RAM nachgekauft und nun einen richtigen 1-MByte-AT. Ein paar Tage später stellt er fest, daß er seinen Rechner auch auf 0 Wait-States einstellen kann. Er hat mittlerweile am Rechner eines Bekannten mit einem Benchmarkprogramm herausbekommen, daß sich die Geschwindigkeit dadurch um etwa 25 Prozent erhöhen läßt.

Der Haken: er bräuchte schnellere RAMs; sowie er den Wartezyklus abschaltet, stürzt sein Rechner ab. Hätte er sich direkt beim Kauf für schnellere RAMs entschieden, so wäre er vielleicht mit 50 DM mehr weggekommen. Jetzt sitzt er auf einem MByte Chips für rund 300 bis 400 DM und müßte denselben Betrag noch einmal für die schnellen RAMs investieren.

Um seinen Frust zu dämpfen, kauft er sich lieber erstmal die langersehnte Festplatte. Er weiß zwar inzwischen, daß in ATs eigentlich sogenannte Kombi-Controller sitzen, die Floppy- und Harddisk-Laufwerke gleichermaßen bedienen, sieht aber keine Notwendigkeit, seinen vorhandenen Floppy-Controller auszumustern. Die mit einem Kombi-Controller gebotene Möglichkeit, auch 1,2-MByte-Disketten zu verarbeiten, verkraftet sein Laufwerk eh nicht.

Er entscheidet sich für eine preiswerte RLL-Platte (30 MByte Kapazität) und zugehörigem Harddisk-Controller für XTs. Für nur wenige Mark mehr, als eine einfache 20-MB-Kombination gekostet hätte, hat er 50 Prozent mehr Kapazität und Geschwindigkeit erstanden. Er hat diese Kombination beim Händler selbst auf einem PC gebenchmarkt und weiß, daß sie schneller war als die normale AT-Ausstattung. Auf seinem AT jedoch – ist die Platte deutlich langsamer als das erwähnte AT-System. (Die technische Erklärung dieses Phänomens können Sie im Abschnitt über Festplatten und Controller nachlesen.)

Doch reingefallen. Aber immerhin, die 30 MByte Kapazität bleiben ihm. Aber auch die Sorge, diese regelmäßig durch Backups auf 360-KByte-Disketten zu sichern – fast 88 Disketten bei voller Festplatte. Der Wunsch nach einem 1,2-MByte-Laufwerk beginnt zu keimen, als er erfährt, daß auf solche HD-Disketten auch doppelt so schnell aufgezeichnet werden kann. Außerdem hat er kürzlich ein Programmpaket für ATs entdeckt, das nur auf HD-Disketten geliefert wird. Jedes Umkopieren auf 360-KByte-Scheiben ist bisher gescheitert, weil die Installation danach nicht mehr funktioniert.

Jetzt stellt sich die große Frage: einen XT-Controller kaufen, der dieses Format unterstützt, und nur den alten Floppy-Controller übrigbehalten oder gleich konsequent einen RLL-Kombi-Controller und beide alten Controller rumliegen haben?

Genug des grausamen Spiels. Der gebeutelte Käufer hat zwar keinen unbrauchbaren Rechner, aber durch Unwissen sehr unglücklich agiert und muß jetzt vielleicht sehr teuer erweitern. Vorher abklären ist die Devise, und die Folgebeiträge sollen Ihnen helfen, dem Händler die richtigen Fragen – ganz besonders im Hinblick auf Erweiterbarkeit – vor dem Kauf zu stellen. Zwar ist ein PC oder AT beziehungsweise ein dazu Kompatibler in der Theorie ein offenes und leicht erweiterbares System, die Praxis enthält aber einige Tücken.

Die ersten Beiträge charakterisieren die wichtigsten Merkmale der verschiedenen PC-Gattungen und geben einen Eindruck von – unserer Meinung nach – in sich stimmigen Systemkonfigurationen. Es schließen sich Artikel an, die sich speziell und technisch etwas tiefergehend mit Auswahlkriterien für die wichtigsten PC-Komponenten befassen.

PC/XTs – 8088 mit Verwandtschaft

Die Einteilung der zur Zeit erhältlichen PCs in Leistungsklassen gründet sich auf den eingesetzten Prozessor. Die älteste Klasse ist der PC beziehungsweise PC/XT, der von IBM seit 1981 stets mit einem 8088 bei 4,77 MHz Takt ausgeliefert wurde. Diese langsamste PC-Variante bildet heute die Basiseinheit für die PC-Performance, das heißt, man gibt die Rechenleistung heutiger Rechner in Vielfachen der ursprünglichen PC-Leistung an.

Mehr Tempo

Eine Form der Beschleunigung stellte die Erhöhung der Taktfrequenz des 8088 auf 8 MHz dar. Alternativ erreicht man etwa die gleiche Leistungserhöhung (Faktor 1,4 bis 1,6) auch mit 4,77 MHz Takt, indem man statt des 8088 mit externem 8-Bit-Datenbus die 16-Bit-Ausführung 8086 nimmt. Ein 8086-Rechner mit 8 MHz bringt es folglich auf rund die doppelte Leistung eines PC/XT.

Einen kleinen Leistungsschub brachten auch die V-Chips von NEC, die anschluß- und funktionskompatibel – also direkt austauschbar – zu 8088 (V20) und 8086 (V30) sind. Die V-Chips benötigen intern weniger Taktzyklen zur Ausführung der meisten Befehle als die Intel-Äquivalente. Bei normalen DOS-Anwendungen gewinnt der Anwender rund 25 Prozent mehr Geschwindigkeit, Multiplikation und Division laufen sogar viermal schneller.

Für die meisten Intel-CPUs gibt es Arithmetik-Coprozessoren (siehe Tabelle), die die reine Rechenleistung der jeweiligen CPU im Umgang mit Fließkommazahlen bei sehr hoher Genauigkeit (80 Bit) um etwa Faktor 10 beschleunigen können. Dazu muß die Anwendersoftware die Coprozessoren (deren speziellen Befehlssatz) auch wirklich benutzen. Die weitaus größte Zahl von DOS-Programmen (wie auch DOS selbst) tut das nicht – und braucht es auch nicht –, so daß der Einsatz eines Coprozessors dann keinen Hauch mehr Geschwindigkeit bringt.

Systemkomponenten

Die System-Leistungsfähigkeit wird keineswegs nur von der CPU, sondern von allen Komponenten bestimmt, wobei für allgemeine Anwendungen das Festplattensystem heute nach der CPU zum wichtigsten Faktor geworden ist.

Ein arbeitsfähiges PC-System beherbergt in seinem Gehäuse zunächst ein sogenanntes Motherboard, auf dem sich CPU, Systemspeicher, Tastaturanschluß und vor allem Steckplätze für Erweiterungen befinden.

Weiter findet man als Einsteckkarte eine Bildschirmansteuerung, mindestens ein Floppy-Laufwerk nebst zugehörigem Controller (Steckkarte) und eine Schnittstelle für den Anschluß eines Druckers. Auch die Stromversorgung (Netzteil) steckt unter der PC-Haube.

Nicht mehr grundsätzlicher Bestandteil, aber fast schon Standard ist eine Festplatte mit entsprechender Controller-Steckkarte, ebenso ist heute eine serielle Schnittstelle etwa zum Anschluß einer Maus üblich.

Außerhalb des Gehäuses gehören dann Bildschirm, Tastatur und Drucker zu den unverzichtbaren Komponenten. Um mit einem PC tatsächlich arbeiten zu können, benötigt man allerdings noch ein Betriebssystem und Anwenderprogramme. An dieser Grundkonfiguration hat sich seit der Erschaffung des PC wenig geändert, wohl aber im Hinblick auf die Leistungsfähigkeit der einzelnen Komponenten.

Low-Cost-PCs

Als Low-Cost-PCs können Geräte mit einer der in der Tabelle als XT-Prozessor bezeichneten CPU gelten, die knapp unter 1000 DM bis knapp darüber kosten. Man bekommt für dieses Geld keine Luxusausstattung, muß also etwa auf eine Festplatte verzichten.

In diesem Bereich kannte man bis vor kurzem noch zwei deutlich unterschiedliche Varianten, nämlich die offenen und die geschlossenen Lösungen. Mittlerweile tritt eine halboffene Mischform zutage, die einige Erweiterungen verhindert oder durch Spezialteile verteuert.

Off Limits

Zu den geschlossenen Systemen zählt alles, was keine Steckplätze für PC-übliche Erweiterungskarten enthält. Diese Rechner als kompatibel zum Industriestandard (sprich:

Fortsetzung auf S. 83

Turbo-Pascal-Praxis

Teil 5

Manfred Zander, Dresden

Im Teil 4 unseres Kurses haben wir uns dem Lesen, Bearbeiten und Schreiben von Dateien anderer Programme gewidmet und Ihnen die ersten Prozeduren für geordnete und ungeordnete Listen vorgestellt. Im Bild 42 (MP 12/89, Seite 370) sind die drei Prozeduren Dazu, Weg und Finde als Listing dargestellt. Wir beendeten den Teil 4 mit der Erläuterung der Prozedur Dazu und setzen nun fort mit den Prozeduren Weg und Finde, bevor wir dann die Arbeit mit Daten in Baumstruktur behandeln.

Das Löschen eines Listenelementes wird mit der Prozedur *Weg* (Bild 42, MP 12/89, S. 370) angewiesen. Der Parameter L zeigt auf die Liste, in der ein Element gelöscht werden soll, der Parameter P zeigt auf das zu löschende Element. Eine Prüfung, ob der Zeiger P wirklich auf ein Element der Liste L zeigt, erfolgt jedoch nicht. Um diese Prozedur übersichtlicher zu gestalten, wurden zwei lokale Pointer vereinbart. Der Pointer N erhält den Zeiger auf den Nachfolger, der Pointer V erhält den Zeiger auf den Vorgänger, des zu löschenden Elementes zugewiesen. Existiert ein Nachfolger, so muß er nach dem Löschen in seinem Feld Vorgänger auf den Vorgänger des zu löschenden Elementes zeigen. Existiert ein Vorgänger, so muß dieser in seinem Feld Nachfolger nach dem Löschen auf den Nachfolger des zu löschenden Elementes zeigen. Wenn das zu löschende Element der Liste keinen Vorgänger besitzt, also das erste der Liste ist, wird dem rufenden Programm über den Parameter L der neue Anfang der Liste mitgeteilt. Nun kann der Speicherplatz, den das aus der Liste gestrichene Element beanspruchte, wieder freigegeben werden.

Zum Suchen eines Listenelementes wurde die Prozedur *Finde* erstellt. Sie hat drei Aufrufparameter. Mit dem Parameter L wird ihr der Zeiger auf das erste Element der zu durchsuchenden Liste übergeben. Im Parameter E steht der Listeninhalt, der gesucht werden soll. Über den Parameter P gibt die Prozedur den Zeiger auf das gefundene Element zurück. Wurde das entsprechende Element nicht gefunden, übergibt die Prozedur den Wert NIL an das rufende Programm. Das Suchen in der Liste erfolgt vom ersten Element an bis der gesuchte Inhalt gefunden wurde, oder aber das Ende der Liste erreicht wurde. Da bei dem gewählten Aufbau der Liste ein Bearbeiten sowohl vom Anfang als auch vom Ende her möglich ist, wird oft die Aufgabe bestehen, das letzte Element zu ermitteln. Dies wird von der Prozedur *Ende* erledigt. Ihr wird über den Parameter L der Zeiger auf ein beliebiges Listenelement übergeben. Mit Hilfe dieses Zeigers läuft sie bis zum letzten Element der Liste und übergibt im Parameter P den Zeiger auf dieses letzte Element der Liste. Die Laufanweisung entspricht der in der Prozedur *Finde* ohne Test auf Identität.

Zur Demonstration der Arbeit mit diesen Prozeduren folgt in Bild 43 ein kleines Programm, welches die beschriebenen Listenoperationen nutzt.

Als Listeninhalt wird wieder der Typ *Integer*

```
Program Trenne_Gerade_Ungerade;

type Listeninhalt = integer;

{$I Liste.INC }

var i,e:integer;
    Heaptop,Lgerade,Lungerade,P:liste;
begin
  Mark(heaptop);
  Lgerade:=NIL;Lungerade:=NIL;
  for i:=1 to 20+random(20) do
    begin
      e:=random(10);
      if (e mod 2 = 0) then Dazu(Lgerade,e)
                       else Dazu(Lungerade,e)
    end;
  Finde(Lgerade,0,P);
  while P<>NIL do
    begin Weg(Lgerade,P); Finde(Lgerade,0,P) end;
  P:=Lgerade;writeln('Die geraden:');
  while P<>NIL do
    begin write(P^.Eintrag:4); P:=P^.Nachfolger end;
  P:=Lungerade;writeln;writeln('Die ungeraden:');
  while P<>NIL do
    begin write(P^.Eintrag:4); P:=P^.Nachfolger end;
  Ende(Lgerade,P);
  writeln;writeln('die geraden rueckwaerts:');
  while P<>NIL do
    begin write(P^.Eintrag:4); P:=P^.Vorgaenger end;
  release(heaptop);
end.
```

***Bild 43** Demonstrationsprogramm für ungeordnete Listen*

vereinbart. Die vereinbarten Pointer-Variablen *Lgerade* und *Lungerade* werden als Zeiger zu zwei unterschiedlichen Listen genutzt, der Pointer P wird als Laufzeiger zu Elementen beider Listen eingesetzt. Zuerst wird den beiden Zeigern Lgerade und Lungerade der Wert NIL zugewiesen; die beiden Listen sind also noch leer. In der folgenden Laufanweisung mit zufälliger Laufgrenze werden Zufallszahlen ermittelt. Die geraden Zufallszahlen werden in der Liste *Lgerade*, die ungeraden Zahlen werden in der Liste *Lungerade* gespeichert. Danach werden alle Elemente der geraden Liste, die den Wert Null enthalten, gelöscht. Im Anschluß daran werden beide Listen auf dem Bildschirm angezeigt. Dabei wird mit dem jeweils ersten Listenelement begonnen. Zur Demonstration der Rückwärtsverarbeitung wird danach der Inhalt der geraden Liste nochmals vom letzten Element an ausgegeben. Die erste und letzte Anweisung im Programm garantieren wiederum, daß alle Reservierungen für dynamische Variablen im Hauptspeicher nach dem Programmlauf wieder freigegeben sind.

Zu den Vor- und Nachteilen dieser Listenorganisation ist folgendes zu sagen. Durch die Einführung eines zweiten Zeigers in den Listenelementen wurden zwei entgegengesetzte Effekte erreicht: Zum einen steigt der Speicherbedarf je Listenelement, zum anderen ist damit die Arbeit mit der Liste wesentlich vereinfacht worden. Durch diese Vereinfachung entstehen vor allem Vorteile bei der Nutzung bzw. Auswertung der Listen, und die Geschwindigkeit der Pointer-Adressierung wird bei dieser Lösung besonders deutlich.

Die beiden vorgestellten Listenstrukturen sind keineswegs die einzig möglichen. Denkbar ist zum Beispiel, bei den geordneten Listen ebenfalls einen Vorgänger-Verweis in den Elementen zu führen, oder aber diesen Verweis bei den ungeordneten Listen wegzulassen. Für welche spezielle Struktur man sich entscheidet, hängt letztendlich vom konkreten Problem ab. Die speziell zu programmierende Aufgabe wird auch entscheiden, welche Vereinfachungen möglich und welche weiteren Listenoperationen erforderlich sind. Weitere Operationen könnten beispielsweise das Einfügen an einer bestimmten Listenoperation, oder auch das Tauschen von zwei Listenelementen sein. Die Programmierung solcher Lösungen ist mit der vorgestellten Methodik leicht möglich.

Bäume

In vielen Aufgabenstellungen sind Daten zu verwalten, die untereinander wohl definierte *Unterstellungsverhältnisse* haben. Als Beispiel kann die Organisationsstruktur eines Betriebes dienen. Ein Betrieb hat genau einen Kopf, den Betriebsdirektor. Diesem Kopf sind die jeweiligen Fachdirektoren untergeordnet. Den Fachdirektoren unterstehen jeweils mehrere Hauptabteilungsleiter, diesen die Abteilungsleiter und so weiter. Bei einer Liste gibt es genau ein erstes und ein letztes Element; sie ist *linear* aufgebaut und damit zur Beschreibung dieser Struktur nicht geeignet. In einem Baum dagegen existiert ebenfalls genau ein erstes Element, es ist aber kein letztes definiert. Jedes Element (oder Knoten) eines Baumes hat genau einen Vorgänger, aber eine beliebige Anzahl von Nachfolgern. So untersteht ein Hauptabteilungsleiter genau einem Direktor, selbst aber unterstehen ihm mehrere Abteilungsleiter. Bäume sind also nicht linear, sondern *verzweigt* aufgebaut. Ein Spezialfall dieser Problematik ist der binäre Baum, den wir zuerst behandeln wollen.

Der binäre Baum

Ein binärer Baum hat genau einen Kopfknoten, der selbst keinen Vorgänger hat. In diesem *Baum-Kopf* sind, wie in jedem anderen Knoten des Baumes auch, jeweils zwei Verweise enthalten, die auf weitere Knoten des Baumes zeigen können. Als Beispiel bietet sich der Stammbaum einer Person an. Der Kopf des entsprechenden Baumes ist die Person selbst. In diesem Kopf gibt es genau zwei Verweise, die auf die Mutter und den Vater zeigen. Die *Knoten* der Mutter und des Vaters wiederum zeigen auf jeweils zwei weitere Knoten, auf die Großeltern. Diese Verweise können, wenn notwendig, immer weiter gefächert werden. Wenn wir den Kopf ei-

nes solchen Baumes als erste Ebene bezeichnen, hat die zweite Ebene, die der Eltern, zwei Einträge, die dritte Ebene kann maximal vier Einträge aufnehmen, die vierte acht und so weiter. Um in einer solchen Struktur von einem beliebigen Knoten aus wieder zum Kopf zu finden, wird in jedem Knoten ein Zeigerwert auf den jeweils übergeordneten gestellt. Damit ergibt sich für ein Knotenelement des binären Baumes folgender Aufbau: Der Eintrag selbst, ein Zeiger zum übergeordneten Knoten, im Beispiel **v**

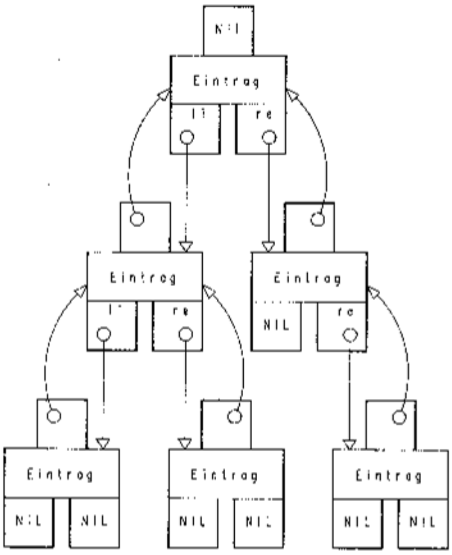


Bild 44 Der binäre Baum

bezeichnet, sowie zwei weitere Zeiger zu den jeweils untergeordneten Knoten, bezeichnet mit links und rechts (Bild 44).
Zur Arbeit mit einer solchen Baumstruktur dienen die in Bild 45 gezeigten Prozeduren. Die Prozedur *Kopfset* organisiert den Beginn des Baumaufbaus; sie reserviert den Speicherplatz für den Kopfknoten und füllt dessen Eintrag. Dann setzt sie alle drei Verweise auf NIL, da dieser Knoten ja noch auf keinen weiteren Knoten zeigt. Über den Parameter L gibt sie dem rufenden Programm den Zeiger auf diesen Kopf des Baumes zurück. Der weitere Aufbau des Baumes erfolgt dann mit der Prozedur *Anfügen*, die die Existenz von mindestens einem Baumknoten voraussetzt. Dieser Prozedur werden in der vorgestellten Realisierung insgesamt fünf Parameter übergeben. Im ersten Parameter R erfolgt die Spezifizierung, ob der neue Knoten rechts oder links angebunden werden soll. Der Zeiger L zeigt auf den jeweiligen Baum, der erweitert werden soll. Die Prozedur hat den Auftrag, einen Knoten mit dem Eintrag E an den Knoten anzubinden, der den Eintrag AE enthält. Über den fünften Parameter P wird nach erfolgreicher Arbeit der Zeiger auf den neuen Knoten zurückgegeben. Hat der Parameter P nach dem Beenden der Prozedur den Wert NIL, konnte der Auftrag nicht erfüllt werden. Das Vorgehen der Prozedur Anfügen kann folgendermaßen beschrieben werden. Zuerst ermittelt sie den Zeigerwert zu den Knoten, an den angefügt werden soll. Dies wird mit Hilfe der Prozedur *Finde* realisiert. Nach Aufruf dieser Suchprozedur enthält die Pointervariable *Tem* den Zeigerwert auf den Knoten mit dem Eintrag AE. Ist dieser Zeiger gleich NIL, so existiert kein Knoten AE im Baum; ein Anfügen ist also nicht möglich. Ist der Knoten, an den angefügt werden soll, vorhanden, wird ermittelt, ob der gewünschte Verweis links oder rechts den Wert NIL besitzt. Dieser Test ist notwendig, da es beispielsweise nicht möglich ist, an einen Knoten zwei rechte Knoten anzubinden. Ist auch dieser Test positiv verlaufen, wird das Anfügen vorgenommen. Dazu wird für den neuen Zweig Platz reserviert, mit dem Eintrag gefüllt wird. Die notwendigen Referenzen in ihm wie auch im übergeordneten Knoten werden ordnungsgemäß gesetzt. Der neue Knoten weist mit **v** auf den alten Knoten zurück und mit **re** und **li** auf NIL. Der alte Knoten weist, wie mit **R** entschieden, entweder mit re oder aber mit li auf den neuen Knoten.
Diese hier vorgestellte Prozedur ist aber nur ein Beispiel. Das jeweils zu lösende Problem wird bestimmen, ob die Tests notwendig sind, und wie viele Parameter wirklich benötigt werden. Wenn von außen garantiert ist, daß alle Konventionen eingehalten werden, ist zum Beispiel die Prozedure *Anf* völlig ausreichend. Ihr wird lediglich der Eintrag E des neuen Knotens und ein Zeiger P auf den Knoten übergeben, an den angefügt werden soll. Der dritte Parameter R spezifiziert wieder in die Richtung, in die weiter angefügt werden soll.
Als letzte soll die Prozedur *Finde* beschrieben werden. Sie hat in ihrer Vereinbarung drei Parameter. Mit dem Zeiger L wird ihr mitgeteilt, in welchem Baum sie suchen soll. Der Parameter E enthält den Eintrag, der gesucht wird, und mit Hilfe des Variablen-Parameters P gibt sie den Zeiger auf den gefundenen Knoten zurück. Enthält der Zeiger P nach dem Beenden der Prozedur immer noch den Wert NIL, so wurde der gesuchte Eintrag nicht gefunden. Durch die rekursive Programmierung dieser Suche konnte die Prozedur sehr einfach gehalten werden. Als erstes testet sie ab, ob der Knoten auf den L zeigt, den gesuchten Eintrag besitzt. Ist dies der Fall, wird P der Zeiger L zugewiesen und die Prozedur ist zu Ende. Im anderen Fall ruft sich die Prozedur selbst auf und setzt dabei den Parameter L auf ihren rechten Verweis. Wurde auch rechts nichts gefunden, erfolgt die weitere Suche links. Mit diesen wenigen Zeilen wird gewährleistet, daß sich die Suche über den kompletten aufgebauten Baum erstreckt. Die Prozedur *Finde* ist ein schönes Beispiel für die Vorteile der rekursiven Programmierung. Im nichtrekursiven Stil wäre diese Prozedur bedeutend komplizierter.
Zur Erläuterung der Nutzung von binären Bäumen erfolgt in Bild 46 wieder die Angabe eines kleinen Hauptprogrammes. Als Eintragtyp für die Knoten des Baumes wurden hier Strings der Länge 20 vereinbart. Zuerst wird der Kopf des Baumes mit dem Eintrag *Kind* gesetzt. Danach wird die Verwandtschaft des Kindes aufgenommen. Dieser Aufbau des Baumes, der hier schrittweise erfolgt, kann in echten Anwendungen natürlich anders organisiert werden. So kann das Wachsen des Baumes aber direkt nachvollzogen werden. Zum Aufbau werden sowohl die Prozedur *Anfügen* als auch die einfache Variante *Anf* genutzt, um die Wirkung beider zu illustrieren. Nach dem Aufbau des Baumes folgt dessen Nutzung. Es wird jeweils ein Name gelesen, und dieser Name wird im Baum gesucht. Existiert ein Knoten mit diesem Eintrag, so wird von diesem aus der Weg

```
(* Bin.INC........Prozeduren fuer binaere Baeume *)
{$A-}
type Richtung = (rechts,links);
     Knoten   = ^Knotentyp;
     Knotentyp= record
                  Eintrag:Eintragtyp;
                  v,li,re:Knoten
                end;

Procedure Kopfset (var L:Knoten;E:Eintragtyp);
  begin
    new(L);
    with L^ do begin Eintrag:=E;v:=NIL;re:=NIL;li:=NIL end;
  end;

Procedure Finde (L:Knoten;E:Eintragtyp;var P:Knoten);
  begin
    P:=NIL;
    if L^.Eintrag = E then P:=L
    else
      begin
        if L^.re<>NIL then Finde(L^.re,E,P);
        if (P=NIL) and (L^.li<>NIL) then Finde(L^.li,E,P);
      end
  end;

Procedure Anfuegen(R:Richtung;L:Knoten;AE,E:Eintragtyp;var P:Knoten);
  var Tem:Knoten;
  begin
    Finde(L,AE,Tem);P:=NIL;
    if Tem = NIL then writeln('Anfuegeknoten nicht gefunden')
      else begin
        if R=rechts then P:=Tem^.re else P:=Tem^.li;
        if P<>NIL then begin P:=NIL;writeln('Anfuegung besetzt') end
          else begin
            new(P);
            if R=rechts then Tem^.re:=P else Tem^.li:=P;
            with P^ do begin v:=Tem;Eintrag:=E;re:=NIL;li:=NIL end
  end end end;

procedure Anf (P:Knoten;E:Eintragtyp;R:Richtung);
  var Tem:Knoten;
  begin
    new(Tem);
    with Tem^ do begin v:=P;Eintrag:=E;re:=NIL;li:=NIL end;
    if R=rechts then P^.re:=Tem else P^.li:=Tem
  end;
```

Bild 45 Prozeduren für binäre Bäume

```
program Stammbaum;

type Eintragtyp = string[20];

{$I Bin.INC }

var Haeptop,L,P: Knoten;    E:Eintragtyp;

begin
  Mark(haeptop);
  kopfset(L,'Kind');
  Anfuegen(links,L,'Kind','Manfred',P);
  Anfuegen(rechts,L,'Kind','Gabi',P);
  Anf(P,'Doris',rechts); Anf(P,'Wilfried',links);
  Finde(L,'Manfred',P);
  Anf(P,'Helga',rechts);    Anf(P,'Heinz',links);
  Anfuegen(rechts,L,'Helga','Auguste',P);
  Anfuegen(rechts,L,'Doris','Marie',P);
  repeat
    write('Ermittelung fuer: ');readln(E);
    Finde(L,E,P);
    while (P^.v <> NIL)and(P<>NIL) do
      begin
        write(' ':20,P^.Eintrag,' ist ');
        if P^.v^.re = P then write('Mutter')
                        else write('Vater');
        writeln(' von: ',P^.v^.Eintrag);
        P:=P^.v
      end
  until E='';
  release(Haeptop)
end.
```

Bild 46 Programm Stammbaum

bis zum Kind zurückverfolgt und protokolliert.
Das Bild 47 stellt den im Programm aufgebauten Stammbaum graphisch dar, und Bild 48 zeigt das entstehende Bildschirmaussehen bei der Bedienung des Programms.

Beliebig verzweigte Bäume

In binären Bäumen kann ein Knoten jeweils nur auf zwei weitere Knoten verweisen. Dies kann natürlich ein Nachteil sein. In den meisten Aufgabenstellungen wird gefordert, daß ein Knoten auf eine beliebige Anzahl von anderen Knoten verweisen kann. Die im folgenden vorgestellte Methodik zur Lösung dieser Aufgabe kann nur *ein* Beispiel sein und sollte auch nicht dogmatisch übernommen werden, wenn eigene Probleme gelöst werden sollen.
Zuerst wird die zu verwaltende Struktur im Hauptspeicher erläutert werden. Im Gegensatz zu den Listen und den binären Bäumen haben wir es nun mit zwei Typen von dynamischen Variablen zu tun. Der erste Typ, bezeichnet mit Knotentyp, beinhaltet die zu verwaltenden Einträge und zwei unterschiedlich typisierte Pointer. Der erste Pointer, bezeichnet mit *vor*, weist auf den jeweiligen Vorgängerknoten. Der zweite Zeiger in jedem Knoten, bezeichnet mit *nL*, zeigt auf eine lineare Liste, in der alle Verweise auf Nachfolgeknoten gesammelt werden. Die Elemente dieser Verweisliste haben je zwei Felder. Das erste Feld, bezeichnet mit *nach*, verweist jeweils auf einen Nachfolgeknoten, und das Feld *next* dient zur linearen Verkettung dieser Nachfolgerliste. Durch die Speicherung der Nachfolgerverweise in einer Liste ist es nun möglich, zu jedem Knoten eine beliebige Anzahl von Verweisen zu anderen Knoten aufzubauen. Damit hat der aufgebaute Baum wieder genau einen Kopfknoten, aber keinen *letzten* Knoten. Jeder Knoten des Baumes, mit Ausnahme des Kopfes, hat genau einen übergeordneten Vorgänger und eine beliebige Anzahl von untergeordneten Nachfolgern

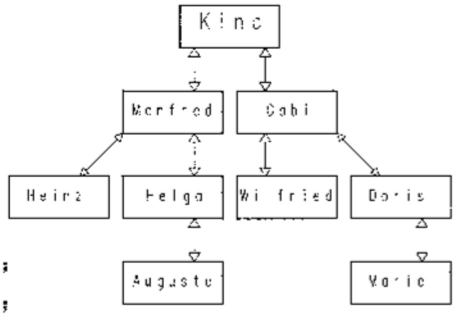


Bild 47 Darstellung des aufgebauten Baumes

(Bild 49). Zur Bearbeitung dieser Baumstruktur im Hauptspeicher werden wieder drei Grundprozeduren vorgestellt. *FindeKnoten* soll nach einem Knoten im Baum suchen. Die Prozedur *NeuerKnoten* dient zum Anhängen eines neuen Knotens an einen bestehenden Baum, bzw. zum Aufbauen eines Baumes, und die Prozedur *Zweigab* dient zum Löschen eines Knotens im Baum. Durch diese Löschprozedur werden aber alle dem zu löschenden Knoten untergeordneten Knoten ebenfalls gelöscht werden! Sowohl die Prozedur *FindeKnoten* als auch *Zweigab* sind rekursiv programmiert (Bild 50).

```
Ermittelung fuer: Auguste
                   Auguste ist Mutter von: Helga
                   Helga ist Mutter von: Manfred
                   Manfred ist Vater von: Kind
Ermittelung fuer: Wilfried
                   Wilfried ist Vater von: Gabi
                   Gabi ist Mutter von: Kind
Ermittelung fuer: Marie
                   Marie ist Mutter von: Doris
                   Doris ist Mutter von: Gabi
                   Gabi ist Mutter von: Kind
Ermittelung fuer:
```

Bild 48 Dialog mit dem Programm Stammbaum

Der Prozedur *FindeKnoten* wird mit dem Parameter B übergeben, in welchen Baum bzw. Baumteil der Eintrag E gesucht werden soll. Der Zeiger auf den gefundenen Knoten mit diesem Eintrag wird dem rufenden Programm über den Parameter P zurückgegeben. Existiert kein Knoten mit dem Eintrag E im durchsuchten Baum, wird über P der Wert NIL übermittelt. Die Identität zweier Einträge wird in dieser Prozedur mit Hilfe der vorausgesetzten Booleschen Funktion *gleich* getestet. Die Suche im Baum selbst ist wieder rekursiv programmiert. Die Prozedur prüft zuerst, ob der erste Knoten des angegebenen Baumes den Eintrag E enthält. Ist dies der Fall, so wird der Variablen P der Zeiger auf diesen Knoten übergeben. Anderenfalls wird diese Prüfung für alle die Knoten organisiert, auf die in der Nachliste des aktuellen Knotens verwiesen wird. Für jeden dieser untergeordneten Knoten ruft sich die Prozedur selbst auf und realisiert so eine vollständige Durchmusterung aller Baumverzweigungen. Diese Suche wird genau dann beendet, wenn entweder der gesuchte Knoten gefunden wurde oder aber der Baum vollständig durchsucht ist.
Die Prozedur NeuerKnoten dient zum schrittweisen Aufbauen des Baumes im Hauptspeicher. Ihre Aufgabe besteht darin, an den Knoten, auf den der Zeiger *AN* zeigt, einen neuen Knoten anzuhängen und in diesem neuen Knoten den Eintrag E abzulegen. Hat der Parameter *AN* den Wert NIL, so handelt es sich um den neuen Kopfknoten eines neuen Baumes, und der Zeiger auf diesen Kopf wird dem rufenden Programm über AN mitgeteilt. Existiert der Knoten, an den angefügt werden soll dagegen, so muß die logische Verknüpfung des alten und des neuen Knotens realisiert werden. Der Zeiger *vor* des neuen Knotens zeigt auf den alten Knoten. Dies ist mit einer einfachen Zuweisung möglich. Für den Verweis des alten Knotens auf den neuen muß die *Nach*liste des alten Knotens um ein Listenelement erweitert werden. Hat der alte Knoten noch keine *Nach*liste, so muß diese eröffnet werden. Somit ist die eindeutige Bindung des neuen Knotens an den Baum erreicht.
Die Prozedur Zweigab hat vordergründig die Aufgabe, den Knoten, auf den der Parameter P zeigt, zu löschen. Dazu muß aber nicht nur der Knoten selbst freigegeben werden. Gleichzeitig ist es notwendig, den Verweis in der *Nach*liste des übergeordneten Knotens zu löschen. Aber auch das genügt noch nicht. Wenn der zu löschende Knoten selbst noch eine Nachliste besitzt, so wären die Knoten, auf die in der Nachliste verwiesen wird, nicht freigegeben, aber vom Baum logisch abgehängt. Es ist also notwendig, nicht nur den Knoten selbst, sondern auch alle untergeordneten Zweige des Knotens zu löschen. Dies ist wieder nur durch eine rekursive Programmierung der Prozedur zu organisieren. Sie geht folgendermaßen vor: Zuerst ruft sie für alle Knoten, die in der Nachliste des zu lö-

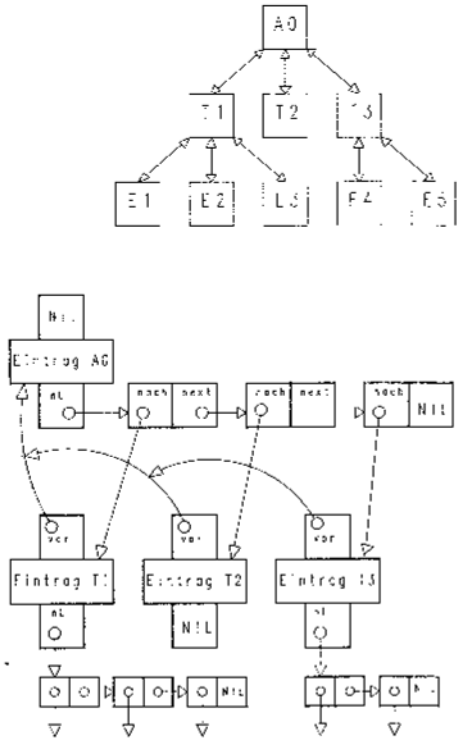


Bild 49 Der beliebig verzweigte Baum (Prinzip und Beispiel)

```
(*******************************)
(* Include Baum.INC            *)
(*******************************)

{$A-}   (* Erzeugen rekursiven Codes !!!! *)
type Knoten     = ^Knotentyp;
     Verweis    = ^Verweistyp;
     Knotentyp  = record Eintrag:Eintragtyp;
                         vor:Knoten;
                         nL :Verweis    end;
     Verweistyp = record nach:Knoten;
                         next:Verweis   end;

Procedure FindeKnoten(Baum:Knoten;E:Eintragtyp;var P:Knoten);
  var V:Verweis;
  begin
    P:=NIL;
    if Baum<> NIL then
      if gleich(Baum^.Eintrag,E) then P:=Baum
      else begin
        V:=Baum^.nL;
        while (V<>NIL)and(P=NIL) do
          begin FindeKnoten(V^.nach,E,P); V:=V^.next
  end end end;
Procedure NeuerAst(var AN:Knoten;E:Eintragtyp);
  var V,L:Verweis;Tem:Knoten;
  begin
    new(Tem);with Tem^ do begin Eintrag:=E;vor:=AN;nL:=NIL end;
    if AN = NIL then AN:=Tem
      else begin
        new(V);with V^ do begin nach:=Tem;next:=NIL end;
        if AN^.nL = NIL then AN^.nL:=V
          else begin
            L:=AN^.nL;
            while L^.next <> NIL do L:=L^.next;
            L^.next:=V
  end   end end;
Procedure Zweigab(Ab:Knoten);
  Var A,V:Verweis;
  begin If Ab<>NIL then begin
    V:=Ab^.nL;
    while V<>NIL do
      begin A:=V^.next;Zweigab(V^.nach);V:=A end;
    if Ab^.vor<>NIL then
      begin
        V:=Ab^.vor^.nL;
        if V^.nach=Ab
          then Ab^.vor^.nL:=V^.next
          else begin
            while V^.nach<>Ab do begin A:=V;V:=V^.next end;
            A^.next:=V^.next end;
        dispose(V) end;
    dispose(Ab)
  end end;
```

Bild 50 ***Prozeduren zur Baumbearbeitung***

schenden Knotens enthalten sind, sich selbst auf. Da bei der Löschung eines untergeordneten Knotens auch der Verweis auf diesen Knoten in der Nachliste gelöscht und freigegeben wird, verschwindet die Nachliste somit vollständig. Sind alle untergeordneten Knoten gelöscht, ist die Nachliste also leer; es muß also nur noch der Verweis auf den zu löschenden Knoten im übergeordneten Knoten beseitigt werden. Dies gilt natürlich nicht für den Kopfknoten eines Baumes. Nachdem alle Verweise auf den zu löschenden Knoten beseitigt sind, kann zuletzt auch er selbst noch freigegeben werden.
Mit Hilfe dieser drei Prozeduren zur Baumbehandlung ist es nun möglich, einen Baum im Hauptspeicher auf- und abzubauen sowie einzelne Knoten im Baum aufzufinden. Im weiteren geht es nun darum, den aufgebauten Baum auszuwerten, das heißt aus dem Baum neue Informationen zu gewinnen. Die dabei möglichen Fragestellungen sind zwar sehr stark abhängig vom Charakter der verwalteten Informationen in den Knoten, es lassen sich aber trotzdem einige allgemeingültige Prinzipien und Fragestellungen herausarbeiten. Beispiele für immer wieder auftretende Aufgaben sind die Ermittlung aller übergeordneten oder auch aller untergeordneten Knoten zu einem konkreten Knoten im Baum.
Das Ergebnis einer solchen Aufgabe ist aber nicht ein einzelnes Objekt, sondern meist eine gewisse Anzahl von Knoten. Das Ergebnis ist also eine Liste von Knoten. Nun wäre es aber zu aufwendig, in dieser Liste die kompletten Knoteninhalte nochmals aufzubauen. Es ist völlig ausreichend, nur die Verweise zu den Knoten, die ermittelt worden sind, in der Ergebnisliste zu vermerken. Im Ergebnis solcher Fragestellungen entstehen also Listen, in denen Knotenverweise gesammelt werden.
Diese Listen entsprechen in ihrem Aufbau völlig den *Nach*listen der Baumknoten. Der Unterschied besteht nur darin, daß die Ergebnislisten zwar auf die Knoten des Baumes verweisen, es aber keinen Rückbezug vom Baum auf diese Listen gibt. Die *Nach*listen geben die Struktur des Baumes an, die Ergebnislisten sind eine Sammlung von Knoten. Es besteht also lediglich ein Unterschied in der Nutzung der Listen, kein Unterschied in ihrem physischen Aufbau. Zu einem Baum können also durch verschiedenste Auswertungen beliebig viele Ergebnis- oder Knotenlisten aufgebaut und gelöscht werden, ohne daß der Baum selbst verändert wird.
Es erweist sich wieder als günstig, die Verwaltung der Knotenlisten speziellen Prozeduren zu übertragen. Diese sind in dem Bild 51 aufgelistet.
Von der Struktur her handelt es sich bei den Knotenlisten um einfach verkettete, ungeordnete Listen. Die verwalteten Listeninhalte sind in diesem Fall Pointer auf Knoten. Alle notwendigen Typvereinbarungen sind bereits in den Baumprozeduren vorhanden, und werden hier vorausgesetzt.
Die Prozedur *DazuKnotenListe* erweitert eine Knotenliste. Der erste Parameter stellt einen Zeiger auf die zu erweiternde Liste dar, der zweite Parameter den aufzunehmenden Knotenverweis. Das neue Listenelement wird an den Anfang der zu erweiternden Liste gesetzt, und der neue Listenanfang wird über den ersten Parameter zurückgegeben. Dieses hier angewandte Prinzip entspricht dem in den ungeordneten Listen.

wird fortgesetzt

```
(*******************************)
(* Include KntenLi.INC         *)
(*******************************)

Procedure DazuKnotenListe(var KnotenListe:Verweis;Kn:Knoten);
  var T:Verweis;
  begin new(T);
    with T^ do begin nach:=Kn;next:=KnotenListe end;
    Knotenliste:=T
  end;
Procedure LoeschenKnotenListe(var KnotenListe:Verweis);
  var T:Verweis;
    begin T:=KnotenListe;
      while T<> NIL do
        begin T:=T^.next;dispose(KnotenListe);KnotenListe:=T end
    end;
Procedure DruckenKnotenListe( KnotenListe:Verweis);
    var T:Verweis;
    begin T:=KnotenListe;
      while T<> NIL do
        begin
          Druckeknoten(T^.nach);
          T:=T^.next;
    end;writeln end;
Function Anzahl(L:Verweis):integer;
  var I:integer;
  begin I:=0;while L<>NIL do begin I:=I+1;L:=L^.next end;Anzahl:=I end;
Function Member(L:Verweis;E:Eintragtyp):boolean;
  begin Member:=false;
    while L<>NIL do begin
      if gleich(L^.nach^.Eintrag,E) then Member:=true;
      L:=L^.next
  end end;
```

Bild 51 ***Prozeduren für Knoten-Listen***

Fortsetzung von S. 78

PC/XT-Prozessoren

8 Bit		16 Bit	
8088	4,77 bis 8MHz	8086	4,77 bis 8MHz
V20	4,77 bis 12MHz	V30	4,77 bis 12MHz
80188	typisch 8MHz	80186	typisch 8MHz
V40	bis 12MHz	V50	bis 12MHz

80188/186 und V40/V50 findet man selten in PCs, sie entsprechen in der Leistung V20/V30 und verfügen über chipinterne I/O-Ports. Alle aufgeführten Prozessoren haben einen maximalen Adreßbereich von 1 MByte.

Der Arithmetik-Coprozessor 8087 verträgt sich mit allen CPUs außer 80188/80186. Zur Zeit gibt es keine Versionen für mehr als 10 MHz. Die CPU darf bei Einsatz eines 8087 nicht schneller getaktet werden, als der 8087 erlaubt.

AT-Prozessoren

16 Bit		32 Bit	
80286	6 bis 20 MHz	80386DX	16 bis 33 MHz
80386SX	16 MHz (bald 20 MHz)	80486	25 bis 33 MHz (geplant 50 MHz)

80286 und 80386SX können maximal 16 MByte physikalisch adressieren, 80386DX und 80486 vier Gigabyte.

Zum 80286 gibt es den Arithmetik-Coprozessor 80287, der auch langsamer als die CPU getaktet werden darf (asynchron). Von Intel sind Ausführungen für maximal 10 MHz erhältlich, von IIT (Santa Clara, USA) 16- und 20-MHz-Versionen. Über spezielle Logik kann ein 80287 auch an 80386DX und SX angebunden werden, üblicherweise verwendet man aber 80387SX (16 MHz) und 80387DX (16 bis 25 MHz verfügbar). Auch die 387er dürfen asynchron zur CPU (± 60 Prozent) getaktet werden. Der 80486 enthält den passenden Coprozessor bereits auf dem Chip.

zum IBM PC) zu bezeichnen, ist fragwürdig. Die Erweiterbarkeit durch Steckkarten beliebiger Fremdhersteller macht schließlich einen wesentlichen Bestandteil eines PC-Systems aus. Vielfach retten sich Hersteller mit einem hauseigenen Erweiterungsbus, an den man eine (hauseigene) Erweiterungsbox (zu hauseigenen Preisen) mit echten PC-Steckplätzen anschließen kann. Die Kosten für solche Zusätze muß man bei einem Preisvergleich mit offenen Systemen also einbeziehen.

Allerdings kann man konstatieren, daß solche Minimal-Rechner für ihren Preis grundsätzlich gut mit Schnittstellen und Grafik ausgestattet sind. Wenn man einen solchen Rechner zum Beispiel bewußt als reinen Schreibmaschinenersatz anschafft und sich sicher ist, daß man etwa ohne Festplatte auskommt, ist solch ein PC in Ordnung.

Die begrenzte Erweiterbarkeit gilt auch für alle uns bekannten Laptops mit 8088-ähnlichen Prozessoren, jedoch sind sie eigentlich als protable Zweitgeräte konzipiert, die gar nicht den Anspruch auf Erweiterbarkeit erheben.

Halboffen

Die Ausstattung der halboffenen XTs entspricht weitgehend der der geschlossenen Systeme, aber diese Rechner bieten zusätzlich Steckplätze. Wenn jedoch Komponenten auf der Hauptplatine durch andere (etwa modernere) auf Steckkarten ersetzt werden sollen, kann es Probleme geben.

Der PC1512 von Schneider möge hier als Beispiel dienen, es gibt auch andere (siehe Tabelle). Bei ihm läßt sich die eingebaute Grafik (CGA, also die schlechteste Grafikqualität für PCs) nicht abschalten. Lediglich eine Hercules- oder MDA-Karte kann in solch einem Fall parallel betrieben werden, eine EGA- oder VGA-Karte hingegen kollidiert adreßmäßig mit der integrierten CGA. Weiterhin ist das Netzteil im Monitorgehäuse eingebaut: Will man einen Hercules-Monitor einsetzen, muß der alte Monitor weiterhin als Netzteil dienen.

Relativ neu sind Harddisk-Controller auf der Hauptplatine, deren Spezial-Interface den Anschluß herstellereigener (meist recht teurer) Festplatten erzwingt. Ein solcher Controller sollte unbedingt wie bei Comodores PC 10/20-III abschaltbar sein. Damit kann man XT-Controller plus Standard-Platte einsetzen, was zusammen billiger als eine herstellerspezifische Platte sein kann.

Offen

Hardwaremäßig sind gerade die Fernost-Produkte diejenigen, die am dichtesten am Original liegen, wenngleich auch hier die ersten Varianten mit „Komplett-Motherboards" auftauchen.

Offene Systeme stellen auch Rechner mit passiver Busplatine dar (nur Busstecker, keine Elektronik). Die gesamte Logik eines XT-Motherboards befindet sich auf einer eigenen Steckkarte. Die vielfach gepriesene Flexibilität dieses Konzepts (einfacher Umstieg auf größere Prozessoren) kommt aber nur zum Tragen, wenn sich dabei die Busbreite des Prozessors nicht ändert – sonst ist ein neues Busboard fällig. Preislich liegen diese Systeme kaum günstiger als Standard-Motherboards, jedoch lassen sich mit PC-Karten halber Länge sehr kompakte Systeme für Spezialgehäuse aufbauen.

Bedingt kompatibel

Zu Beginn der PC-Ära achteten Nachbauer nicht so genau auf die Hardware-Details des PCs. Das rächte sich bitter, als immer mehr Programmierer am DOS und ROM-BIOS vorbei die Hardware des IBM PC direkt manipulierten, damit ihre Programme schneller und folglich attraktiver wurden. Inzwischen ist quasi kein PC-Port oder Speicherbereich vor dem direkten Zugriff eines Programmierers sicher, und wehe, Drucker-Port oder Bildspeicher weichen vom Original ab! Derartige PCs gab es einige (siehe Tabelle), aber man findet sie wohl nur noch im Secondhandshop.

Systemunterschiede bei XT-Systemen

Offen
Hauptplatine enthält nur CPU, Coprozessor-Sockel, Arbeitsspeicher, Tastaturanschluß, Steckplätze für Erweiterungen. Ein weitergehender Ausbau stößt nicht auf Probleme mit vorhandenen Komponenten.

Typische Vertreter: IBM PC, Fernost-Nachbauten

Halboffen
Hauptplatine enthält zusätzlich Floppy-Controller, Video-Logik, parallele und serielle Schnittstelle, mitunter auch Harddisk-Controller. Probleme entstehen bei Erweiterungen, wenn die Komponenten auf der Hauptplatine durch technisch verbesserte Ausführungen ersetzt werden sollen, die On-board-Komponenten aber nicht abschaltbar sind. Beim Schneider EuroPC sind die Abschaltmöglichkeiten vorbildlich gelöst (per Setup), allerdings hat er nur einen Slot für PC-Karten halber Länge. Bei Commodore PC 10/20-III stören der nicht abschaltbare Floppy-Controller, bei Amstrad 1512/1640 vor allem Grafik und Floppy-Controller.

Typische Vertreter: Amstrad/Schneider PC 1512/1640, Commodore PC 10/20-III, Schneider Euro PC

Geschlossen
Geschlossene Systeme enthalten meist alle Komponenten wie bei halboffen aufgeführt, haben jedoch überhaupt keine Steckplätze, sondern allenfalls Spezialbusse zum Anschluß von Erweiterungs-Boxen für externe Steckplätze. Erweiterungen sind gar nicht oder nur relativ kostspielig (herstellerspezifische Lösungen) möglich.

Typische Vertreter: Commodore PC-1, Atari PC 1 (beide werden nicht mehr hergestellt), Zenith Easy PC, fast alle Laptops

Bedingt kompatibel
Bedingt kompatibel sind nicht nur die geschlossenen Systeme. Einige ältere PCs weichen in wichtigen Hardware-Details vom PC-Standard ab (spezielle Grafik, andere Floppy-Formate oder Controller-Chips, nicht PC-gemäße Adreßlage wichtiger System-Ports und mehr). Das kann dazu führen, daß bestimmte PC-Software nicht läuft, vom Rechnerhersteller maßgescheiderte DOS-Versionen nötig werden (die möglicherweise nicht aufdatiert werden) oder nur herstellerspezifische Erweiterungskarten verwendbar sind.

Typische Vertreter: DEC Rainbow, Olivetti M24, Victor Sirius, die ersten Siemens PCDs, Zenith 150-Reihe

Anwendbar

Auch wenn der XT von High-Tech-Fans nur noch milde als Schnecken-PC belächelt wird, so reicht seine Leistung für sehr viele Zwecke aus. So wie man mit einem Kleinwagen auch jedes Ziel erreichen kann, kann man alle Aufgaben, für die es DOS-Programme gibt, letztlich damit lösen. Es kann schlimmstenfalls sehr langsam zugehen oder eng im Speicher werden.

Die sinnvollste Anwendung liegt im Bereich einfacher Textverarbeitung, wo also zum Beispiel *keine* Vollgrafik wie bei MS-Word ins Spiel kommt. Die Entwicklung mittlerer Programme (bis 1000 Zeilen) geht sogar recht zügig von der Hand, sei es in Hochsprache oder Assembler. Bei Interpreter-Sprachen wie GWBASIC wird's schon langweiliger.

Für DFÜ (Datenfernübertragung) ist ein solcher Rechner hingegen keineswegs überfordert (Mailbox-Zugang oder Btx), selbst Datenbankanwendungen, Buchführung oder ähnliches lassen sich sehr wohl damit betreiben. Allerdings wird dann eine Festplatte mit 20 oder 30 MByte Kapazität zur Bedingung. Diese Plattenkapazität setzt letztlich auch die Grenze des Sinnvollen: Wenn Anwendungen mehr verlangen, dann ist meist auch der nächstgrößere Rechner gut ausgelastet.

Definitiv nicht zugänglich sind den PCs jedoch Betriebssysteme für den Protected Mode des 80286, also etwa OS/2 oder die meisten Unix-Abkömmlinge. Minix zum Beispiel gibt es jedoch auch für XTs.

PC-Lösung

Eine preis-/leistungsmäßig ausgewogene und ergonomisch sinnvolle PC-Konfiguration sollte nur mit Monochrom-Monitor und Hercules-Grafik ausgestattet werden, die zusammen schon für 250 bis 300 DM zu haben sind. Eine preislich vergleichbare CGA-Lösung (Color Graphics Adapter) ist restlos veraltet und reiner Augenstreß.

Eine Kombination aus EGA-Karte und scharf justiertem Farbmonitor (zusammen rund 1000 DM) ist unterstes Level für Farbausstattung. Im Textmodus belastet EGA den Prozessor nicht mehr als bei Hercules oder CGA, bei echtem Grafik-Betrieb etwa unter GEM oder Windows kommt ein 4,77-MHz-8088 aber schon „ins Schwitzen". Die nächsthöhere Bildqualität VGA (auch monochrom schon rund 1000 DM) steht eigentlich nicht mehr im rechten Verhältnis zu Leistung und Preis eines XTs, ist aber wegen der höheren Auflösung für die Augen die beste Wahl.

Als Massenspeicher kann man heute nur noch RLL-Systeme empfehlen, da man hiermit für nur wenige Mark mehr als die preisgünstigsten MFM-Systeme (20 MByte) das 1,5fache an Datentransfergeschwindigkeit und Kapazität bekommt.

Als Floppy-Laufwerke sind bei XTs 360-KByte-Versionen für 5,25 Zoll der Standard, es lassen sich aber 3,5-Zoll-Drives (720 KByte) an einem einfachen XT-Controller be-

treiben [1]. High-Density-Laufwerke für 1,2- und 1,44-MByte-Scheiben erfordern einen speziellen Floppy-Controller: Der muß ein eigenes ROM-BIOS zur Bedienung dieser Formate haben und hardwaremäßig mit der doppelten Datenrate eines XT-Controllers schreiben und lesen können. Bei den Tastaturen dominiert heute das MF-2-Layout. MF-2 hat den Segen des DIN und ist nicht nur der neue, sondern überhaupt der erste Standard auf diesem Gebiet. Damit gehört die Suche nach Sonderfunktionen bei der Arbeit an verschiedenen PCs langsam der Vergangenheit an. Allerdings sollte man beim Nachrüsten älterer PCs mit MF-2-Tastatur vor dem Kauf prüfen, ob der Treiber des verwendeten DOS und die Beschriftung auf der Tastatur exakt zusammenpassen. Andernfalls verschiedene Treiber oder Tastaturen ausprobieren.

[1] W. Götz, D. Grell, XT wird doppelspurig, 3,5- oder 5,25-Zoll-Drives mit 80 Spuren, c't 7/89, S. 216 ff.

Swapping kann's oft nicht schnell genug sein.

8-Bit-Karten (XT-Controller) sind im AT ganz besondere Systembremsen und sollten nur als Notlösungen eingesetzt werden. Aber auch die üblicherweise mitgelieferten MFM-Kombi-Controller gehören heute zum alten Eisen, auch wenn deren maximale Datentransferrate mit 240 KByte/s durchaus für viele Anwendungen ausreicht.

Für vergleichsweise wenige Mark mehr (rund 100 DM) erhält man jedoch schon RLL-Kombi-Controller mit Interleave 1:1, die etwa 600 bis 700 KByte/s erreichen, allerdings auch RLL-taugliche Platten erfordern, die ihrerseits auch geringfügig teurer als MFM-Versionen sind. Damit nähert man sich dem Leistungsbereich der noch relativ teuren ESDI- oder SCSI-Systeme.

Die niedrigste Kapazität der heute in ATs angebotenen MFM-Festplatten liegt bei 20 MByte, 40 MByte gelten aber als angemessen. In dem Moment, wo man sich für ein RLL-System entscheidet, erhält man bei quasi baugleichen Platten (lediglich mit höherwertiger Oberflächenbeschichtung) 30 beziehungsweise 60 MByte Kapazität. 60 MByte sind gerade richtig, wenn man zum Beispiel OS/2 und DOS gleichzeitig auf der Platte hält und keine Riesenanwendungen fährt.

Weiterhin Stand der Technik in einem heutigen AT sind ein 5,25-Zoll-Laufwerk (360 KB/1,2 MB) und ein 3,5-Zoll-Drive (720 KB/1,44 MB). Damit lassen sich alle in der PC-Welt gängigen Disketten verarbeiten.

Die preiswerteste und auch für ATs durchaus tolerable Video-Minimal-Ausstattung ist Hercules-Karte und Monochrom-Schirm. Man kann hier aber auch bedenkenlos EGA oder VGA einsetzen, CPU-Power ist auch für Grafik genug da. Am sinnvollsten – mit Blick in die Zukunft, etwa auf OD/2-Software – ist heute sicher eine VGA-Karte mit 16-Bit-Bus zusammen mit einem Multinorm-Farbmonitor. Eine solche Lösung macht mit rund 2000 DM allerdings einen erklecklichen Anteil an den Gesamtkosten aus.

Die zweitgünstigste, aber immer noch zu-

Der schnellere PC heißt AT

80286-CPU und Advanced Technology

1984 brachte die IBM mit dem AT (Advanced Technology) ihren ersten 80286-Rechner (6 MHz Takt, 1 Wait-State bei RAM-Zugriffen) heraus, der dreimal so schnell wie der erste PC war. Später hat IBM die Taktfrequenz auf 8 MHz (1 Wait) gesteigert, in den Modellen 50 und 60 gar auf 10 MHz und 0 Wait.

Den Stand der 80286-Technik repräsentieren jedoch die Nachahmer. Harris beziehungsweise AMD stellen mittlerweile 16- und 20-MHz-Versionen der 80286 her, 25-MHz-Varianten sind avisiert. Ein 10-MHz-AT (0 Wait-States) kann es auf etwa die siebenfache, ein 20-MHz-AT auf die 14fache Leistung eines PC bringen.

Allerdings hat ein AT gegenüber einem PC auch andere Vorteile, zum Beispiel verfügt er über Steckplätze mit 16 Bit Busbreite. Das wichtigste ist wohl, daß die 80286-CPU bis zu 16 MByte Speicher adressieren kann. Jedoch kann sie den Speicher oberhalb von 1 MByte nur in einer Betriebsart nutzen, in der DOS und DOS-Programme nicht lauffähig sind. Lediglich als RAM-Disk oder Hardisk-Cache ist dieser Speicher auch von DOS nutzbar. Ergo: Für DOS-Benutzer ist ein AT nichts anderes als ein schneller PC, der Speicherreserven für eine RAM-Disk mitbringt.

Erst Betriebssysteme wie OS/2 oder Unix können das gesamte Speicherangebot in einem AT direkt ausschöpfen. Einige Unix-Varianten bieten sogar – ähnlich wie OS/2 – die Möglichkeit, als eine Task auch DOS (und darin DOS-Anwendungen) laufen zu lassen. Allerdings gibt es bei letzterem immer einige systembedingte Einschränkungen: Wenn 100prozentige DOS-Kompatibilität gefordert ist, muß man DOS pur fahren.

AT-System

So viele Varianten wie vom PC existieren bei ATs nicht, und Inkompatibilität haben die Nachbauer nach den negativen PC-Erfahrungen von vornherein vermieden. So ist vornehmlich darauf zu achten, daß ein ins Auge gefaßter AT nach dem Stand der Technik ausgestattet ist. Auch von billigsten Produkten kann man heute verlangen, daß sie mit 10 MHz Takt und 0 Wait-States beim RAM-Zugriff daherkommen. Für DOS-Betrieb sind in der Basisausstattung 1 MByte RAM üblich und ausreichend, und zwar in der Aufteilung 640 KByte für DOS, 384 KByte Extended Memory.

Unter OS/2 oder Unix sind vier MByte kein Luxus. Bei RAM-Knappheit beginnt OS/2 nämlich (wie auch Unix), dauernd nicht benötigte Programmteile auf die Platte auszulagern und nachzuladen (sogenanntes Swapping). Da können leicht Situationen entstehen, wo selbst schnelle Harddisk-Systeme das Arbeiten zur Qual werden lassen.

Die Geschwindigkeit des Harddisk-Systems ist für die Systemleistung eines Rechners fast genauso wichtig wie die Rechenleistung der CPU. Vor allem bei der Verwaltung großer Datenbestände (Datenbanken, Server in Netzwerken) oder beim erwähnten

Es gibt auch noch was anderes

Rechner mit den 680x0-CPUs von Motorola sind ernstzunehmende und leistungsfähige Rivalen zu Computern mit Intel-Prozessoren. Der Begriff Personalcomputer, also der persönliche Rechner, war zu Beginn nicht so zwanghaft mit dem Namen IBM verknüpft wie heute. In seinem ursprünglichen Sinne zählen deshalb Apple Macintosh, Atari ST und Amiga auch dazu. Jeder der genannten Rechner ist für einen ganz definierten Anwendungszweck zur Zeit *die* Wahl vieler Profis.

Macintosh

Die Macintosh-Modelle litten lange unter einer Hochpreispolitik, die ihrer Leistung nicht angemessen war. Das hat sich in jüngster Zeit deutlich geändert. Hinzu kommt, daß der Mac-Anwender Zugriff auf eine über viele Jahre gewachsene, ausgereifte und in der Bedienung einheitliche Software hat – davon kann die gesamte übrige Rechnerwelt nur träumen.

Seit es den Mac II mit Einsteckplätzen gibt, besteht auch Zugang zu sehr leistungsfähiger Peripherie (vor allem Grafik), und der neueste Mac IIci etwa (mit 25-MHz-68030) darf sich mit ebensolcher Berechtigung „Workstation" nennen wie ein AT mit 33-MHz-80386. Bei einem Preis von voraussichtlich knapp 20 000 DM gehört er natürlich in die gehobene Profi-Klasse, ist aber zum Beispiel preiswerter als die IBM-Modelle 70 oder 80 mit 20-MHz-80386.

Großer Beliebtheit erfreut sich die Mac-Familie schon lange im Lager der Desktop-Publisher. Wer absehen kann, daß professionelles DTP die dominante Aufgabe für seinen Rechner sein wird, sollte unbedingt einen intensiven Blick über den Kompatibilitätszaun werfen. Allerdings wird er als Profi nicht mit der Standardgrafik leben können, sondern muß – ebenso wie bei einem Rechner mit Intel-CPU – rund 5000 bis 10 000 DM für einen Spezial-Grafikadapter nebst Monitor zusätzlich veranschlagen.

Atari ST

Weniger kostenintensiv geht's bei Atari zu, aber auch die STs haben eine – in Deutschland sogar unbestrittene – Domäne: Musikelektronik. Wer schwerpunktmäßig Synthesizer programmieren will, es bei Textverarbeitung gern schwarz auf weiß ohne Augenschaden mag und keine allzu exotischen Sonderwünsche an verfügbarer Software hat, für den ist die ST-Familie die richtige und eine preisgünstige Wahl.

Amiga

Beim Amiga ist es zwar ratsam, ihn für täglich mehrstündiges Arbeiten aus gesundheitlichen Gründen mit einer weniger flimmernden Grafikausstattung zu versehen, jedoch ist seine flimmernde Verwandtschaft zum heimischen Fernseher auch seine große Tugend: wenn's um häusliche Video-Produktionen (Betiteln und ähnliches) geht, dann sind die für den Amiga verfügbaren Lösungen in Hard- und Software preisgünstig und leistungsfähig.

Dennoch ist vor der Anschaffung von Rechnern, die nicht auf Intel-Chips basieren, zu klären, ob nicht irgendwelche Randbedingungen (Ausbildung, Arbeitsplatz) die Anschaffung eines PC-Kompatiblen nahelegen. Zwar gibt es für den Amiga 2000 PC- beziehungsweise AT-Einsteckkarten und für den Atari ST mittlerweile sogenannte PC-Emulatoren, man sollte sich aber dennoch damit abfinden, daß der Kauf eines solchen Gerätes die Abkehr vom großen Standard darstellt. Emulatoren sind naturgemäß kompromißbehaftete Nachahmungen, und wenn absolute Kompatibilität ganz oben auf der Prioritätenliste steht, sind diese Rechner zweite Wahl.

kunftsorientierte Lösung bestünde in einer VGA-Karte mit Monochrom-Schirm (1000 bis 1200 DM). Ein EGA-Farbsystem ist mit etwa 1000 DM nur halb so teuer wie eine volle VGA-Ausrüstung, stellt aber mit ihrer standardmäßig geringeren Auflösung nur die zweite Wahl dar. Gelegenheits- und Hobby-Computeristen sind damit aber gut bedient und müssen nicht befürchten, ihr Augenlicht über Gebühr zu strapazieren.

ATs kann man natürlich noch weitaus schneller (16 MHz, 20 MHz) bekommen, aber sie kosten dann auch mehr. Verfallen Sie vor allem als Hobby-Anwender nicht allzusehr dem Geschwindigkeitsrausch: Steigerungen um Faktor zwei werden Sie nämlich nur bei Aufgaben spüren, wo ausschließlich CPU-Arbeit gefragt ist. Bei den meisten Anwendungen wartet die CPU auf Ihre Tastendrücke.

Interessanter ist bei diesen schnellen (und modernen) Versionen vielmehr, daß sie meist mit dem NEAT-Chip-Satz ausgerüstet sind. Dadurch kann man unter DOS den Speicher besser ausnutzen (residente Programme aus dem RAM unterhalb 640 KByte heraushalten) oder beispielsweise die Video-Ausgabe durch Verlagern eines 8 Bit breiten VGA-ROM-BIOS ins schnelle 16-Bit-RAM (Shadow-RAM) deutlich beschleunigen.

Eignung

Selbst mit einem relativ karg ausgestatteten AT kann man sich – erst als Hobby-Anwender – an die meisten DOS-Anwendungen wagen. Auch Leistungsfresser wie CAD, DTP, größere Datenbankanwendungen, Tabellenkalkulation oder wissenschaftliche Berechnungen, die auf einem PC zur Qual werden können, sind in vernünftigen Zeiträumen zu bewältigen. Als Netzwerkserver sollte ein 80286-Rechner aber nur in kleinen Netzen (drei, vier User) eingesetzt werden.

Die Anforderungen der genannten Anwendungen lassen sich zwar schlecht quantifizieren, aber ein rundum gut ausgestatteter AT für 20-MHz-80286 reicht für die meisten Fälle auch im professionellen Einsatz aus. Vor allem ist die zur Zeit schnellste erhältliche Alternative, ein 80386-AT mit 33 MHz, gar nicht so viel schneller wie gemeinhin angenommen wird. Will man es zudem in Zukunft bei OS/2, dem optimal auf den 80286 abgestimmten Betriebssystem belassen, ist der Zwang zu 80386-CPU allerdings eine weniger glückliche Wahl, doch zu alldem mehr im Abschnitt über die nächste AT-Generation.

Jenseits des einfachen AT

Rechner mit 32-Bit-CPUs 80386 und 80486

Mit der Schaffung der 32-Bit-CPU 80386 hat sich das bislang gezeichnete Bild zunächst nicht gewandelt: Der DOS-User sitzt an einem schnelleren AT mit 640 KByte Arbeitsspeicher. Zwar hat IBM das Hardwarekonzept bei seinem ersten 80386-Rechner Modell 80 insofern geändert, als man – wie auch bei den 80286-Maschinen Modell 50 und 60 – ein inkompatibles Bussystem namens Mikrokanal einführte, aber grundsätzlich ist es ein AT. Inwiefern das Bussystem als Entscheidungskriterium für die Anschaffung eines Rechners Bedeutung erlangen kann, behandelt ein eigener Kasten.

Kein Raser

Bei gleichem Takt und gleicher Anzahl Wait-States ist ein 80386-Prozessor kaum mehr als 20 Prozent schneller als ein 80286. Jedenfalls nicht mit DOS-Software, die – für 8088 geschrieben – auf seine Fähigkeiten (und Problemchen) keine Rücksicht nimmt. Der 80386-Chip ist allerdings mit 33 MHz erhältlich (80286 maximal 20 MHz).

Ab 25 MHz sind die Buszyklen jedoch so knapp bemessen, daß die RAM-Ansteuerung zum Problem wird. Entweder muß der Hersteller sehr schnelle und sehr teure RAMs spendieren, 1 Wait-State einlegen (etwa 20 Prozent weniger Leistung) oder aber zu einer Cache-Lösung greifen. Ein Cache-System wirkt trotz langsamer RAMs im Hauptspeicher, als käme es ohne Wartezyklen aus. Ein Cache-Controller achtet darauf, daß sich alle gerade benötigten Befehle und Daten in einem sehr schnellen Extra-Speicher ohne Wait-States befinden.

Speicherkünstler

Der 80386-Chip bringt – unabhängig von der tatsächlichen Gestaltung der Hardware – eine neue qualitative Dimension ins PC-Geschehen. Per Software ist ohne bauliche Veränderungen eine völlige Umstrukturierung des gesamten adressierbaren Speichers möglich. So lassen sich Löcher im Speicherbereich oberhalb des Videoadapters mit RAM auffüllen oder überflüssige ROMs ausblenden – was man sonst nur auf 80286-NEAT-Rechnern vermag.[1] Spezielle 80386-Software (etwa DESQview) ermöglicht die Einrichtung mehrerer DOS-Speicherbereiche von 640 KByte, zwischen denen (und den darin laufenden Anwendungen) sich per Tastendruck umschalten läßt: In gewisser Weise überwindet man so die 640-KB-Barriere für DOS. Daß man jetzt Speicherbereiche bis zu vier Gigabyte linear adressieren kann, ließ vor allem die Unix-Anbieter stürmisch vom 80286 zum 80386 überlaufen.

Dem 80386DX (so der volle Name der 32-Bit-Version) hat Intel den preiswerteren 16-Bitter 80386SX zur Seite gestellt. Dieser rangiert leistungsmäßig rund zehn Prozent unterhalb des 80286 bei gleicher Taktfrequenz, die beim SX noch bei maximal 16 MHz liegt. Dennoch steht der SX – mit allen inneren Vorzügen des 80386DX bestens gewappnet – in direkter Konkurrenz zum 80286 und gilt langfristig als dessen Ablösung.

80486

Der PC ist noch schneller geworden und wird jetzt gern als Unix-Plattform oder Workstation bezeichnet. Die markantesten Wesenszüge des 80486 sind ein integrierter Arithmetik-Coprozessor nebst 8 KByte großem Cache und Intels Ankündigung, diesen Chip eines Tages mit einer Taktfrequenz von 50 MHz fertigen zu wollen. Einen tatsächlich lieferbaren Rechner mit 80486 (noch bei 25 MHz) bietet momentan nur IBM in Gestalt des Modell 70-A21 für etwas über 30 000 DM an.

Ohne Nutzung des internen Arithmetik-Teils ist der 80486 bei gleichem Takt etwa 50 bis 80 Prozent schneller als der 80386. Bei Anwendungen für den Coprozessor soll das integrierte Gespann des 80486 aber erheblich schneller als ein Gespann aus 80386/387 sein.

80386SX kontra 80286

80386SX- und 80286-AT liegen leistungsmäßig so dicht beisammen, daß man die übrige

[1] NEAT – New Enhanced AT; PCs, die auf der Grundlage des NEAT-Chipsatzes von Chips & Technologies aufgebaut sind, der bis zu 25 MHz erreichen soll. (Red. MP)

ISA, EISA und Mikrokanal

Der Mikrokanal ist technisch sicher besser gelungen als der PC- beziehungsweise AT-Bus, der mittlerweile ISA (Industrie-Standard-Architektur) heißt. Jedoch koppelte er den Anwender anfangs völlig vom gesamten bis dahin verfügbaren Steckkartenangebot ab. Nach rund zweieinhalb Jahren hat sich die Lage allerdings deutlich gebessert, und da die PS/2-Steckkarten für den Mikrokanal auch auf IBMs angekündigten RISC-Maschinen laufen sollen, gewinnt der Mikrokanal vielleicht in ein, zwei Jahren eine größere Bedeutung.

Heute ist zwar schön, daß es da einen Bus gibt, auf dem mehrere Prozessoren parallel arbeiten können oder an den man intelligente Schnittstellenkarten (mit eigenen Prozessoren oder DMA-Chips) anschließen kann. Ein Zwang zum Übergang auf den neuen Bus – vor allem für den einzelnen Endanwender – kann aber nur bestehen, wenn dieser attraktive Lösungen bietet, die zum unverzichtbaren Standard avancieren und sich mit dem Standard-Bus nicht realisieren lassen. Das kann der Mikrokanal heute aber genausowenig bieten wie EISA.

Mit der erweiterten Industrie-Standard-Architektur EISA haben die konservativen Computerhersteller nach langem Lamento zugegeben, daß der Mikrokanal was hat, was der AT-Bus nicht hat. Bei vergleichbaren Leistungsmerkmalen zum Mikrokanal hat man mit einem geschickten Steckerkonzept erreicht, daß im EISA-Bus auch normale ISA-Karten laufen. So jedenfalls die Theorie. Bisher haben wir keine technischen Erfahrungen sammeln können, denn noch hat kein lauffähiger EISA-Rechner den Weg in die Redaktion gefunden.

Aber sei's drum, ehe attraktive, unverzichtbare Steckkarten für EISA auf den Markt kommen, wird wahrscheinlich noch mehr Zeit ins Land gehen, als das für den Mikrokanal der Fall sein wird. Zwar wird man wohl mit dem Kauf eines Rechners mit EISA nichts falsch machen können, ob es hingegen heute schon einen höheren Rechnerpreis rechtfertigt, ist sehr fraglich.

Selbst sogenannte Power-User, die immer vorn an der Entwicklungsfront mit dem jeweils teuersten und schnellsten Equipment mitkämpfen müssen, versäumen zur Zeit nichts, wenn sie sich einen ISA-Rechner kaufen. Spätestens in drei Jahren kaufen sie sich sowieso den nächsten Rechner, und bis dahin sieht man klarer.

Rechnerausstattung nach denselben Gesichtspunkten wählt. Sollte aber eine gelegentliche Beschäftigung mit Unix auf ihrem zukünftigen Programm stehen, dann ist ein 80386SX eindeutig die bessere Wahl, denn schon jetzt ist eine Vernachlässigung der Unix-Versionen für 80286-Maschinen spürbar. Auch wer gern DOS-ähnliche Multitasking-Umgebungen (Windows/386, DESQview) erproben möchte oder schon heute auf OS/386 spekuliert, ist mit einem 80386SX-System gut beraten. Eine neue interessante Anwendungsmöglichkeit, nämlich etliche MByte große Datenfelder unter DOS zu bearbeiten, bieten auch nur 80386er.

Profi-Liga

Ein 80386-AT für den Hausgebrauch ist Luxus. Privat hält auch ein SX-AT selbst dem Anspruchsvollen alle Optionen auf die Zukunft offen. Sinnvoll ausgelastet sind DX-Maschinen nur in professionellen Umgebungen bei Anwendungen, wo selbst ein 20-MHz-80286 das gutbezahlte Personal in Koffeinvergiftungen treibt.

Wenn Anwendungen laufen, die sich mit Coprozessoren deutlich beschleunigen lassen, bringt der Übergang von 80286/80287 zu 80386/387 einen erheblichen Leistungsschub, denn als Gespann liegen 386/387 deutlich besser als ihre Kollegen. Eventuell lohnt sogar das Umstellen auf den noch schnelleren (und etwas teureren) Weitek-Coprozessor beziehungsweise den 80486.

Ferner sind ein RLL-1:1-Controller und eine 65-MByte-Platte hier als untere Grenze anzusehen. ESDI-Systme sind die nächstbessere Wahl, aber SCSI ist vor allem dann ins Auge zu fassen, wenn die Plattengröße weit über 100 MByte liegt. Dann ist nämlich auch ein Streamer-Laufwerk (Bandgerät) nötig, um die Datenbestände regelmäßig zu sichern, und da bleibt eine SCSI-Lösung sozusagen in der Systemfamilie.

Unter Unix oder OS/2 sind vier MByte RAM das Profi-Minimum, eine Platte mit mehr als 100 MByte allein für Unix ist ebenfalls empfehlenswert. Eine 16-Bit-VGA sollte nicht fehlen, aber es gibt sicher genügend Einsatzbereiche, wo man mit einer monochromen Hercules-Ausstattung auskommt. Bei professionellem CAD oder DTP ist jedoch hochwertiges grafisches Spezial-Equipment gefordert, das nicht selten mehr als der restliche Rechner kostet.

Die avisierten 486-ATs kann man derzeit mangels genauerer Kenntnis nur als die lineare Fortsetzung der 386er ansehen, entsprechend ausrüsten und einsetzen. Aber man darf sicher sein, daß auch der 50-MHz-80486 bald seinen Meister in Gestalt eines 80586 findet.

Nachdruck aus c't – Magazin für Computertechnik, Heft 1/1990

Teil 2 dieses Beitrages erscheint in MP 4/1990 und befaßt sich mit den Erweiterungsmöglichkeiten und Grafikstandards. *MP*

UeberAll das gleiche

DOS-Befehle in allen Directories

Michael Bach, Karl-Friedrich Probst

*Wer nur eine kleine Festplatte in seinem PC installiert hat, kann schon mal mit dem Problem Platte voll konfrontiert werden. Da hilft nur eins – unter den Daten gehörig aufräumen. Dabei stellt man möglicherweise fest, daß in mehreren Verzeichnissen überflüssige Dateien (insbesondere *.Bak-Dateien) vorhanden sind. Zum Löschen der Dateien käme ein kleines Programm gerade recht, das diese Arbeit dann UeberAll erledigt.*

Es muß also ein Programm her, das jeden eingegebenen DOS-Befehl im aktuellen und in allen tieferliegenden Unterverzeichnissen nacheinander ausführt. UeberAll ist ein Programm, das diese Bedingung erfüllt. Es ist vollständig in Turbo-Pascal 5.0 geschrieben und auf allen MSDOS-Rechnern lauffähig.

Aufrufen wird es mit UA und anschließenden Parametern. Bei fehlenden Parametern bricht das Programm ab und gibt eine dementsprechende Fehlermeldung aus. Bei ordnungsgemäßem Aufruf wird zur Sicherheit nochmal gefragt, ob der Befehl auch wirklich ausgeführt werden soll. Beantwortet man diese Frage mit J oder j, beginnt das Programm mit seiner Arbeit im gerade aktuellen Verzeichnis. Um sich mit dem Programm vertraut zu machen, sollte man am Anfang harmlose Befehlsfolgen wie „UA dir *.bak" (alle BAK-Dateien auflisten), oder mit UA dir |sort (alle Einträge sortiert ausgeben), eingeben. Vorsicht ist zumindest bei DEL, ERASE oder ähnlichen Befehlen angebracht, denn schnell ist ein Programm oder eine Datei, auch bei mehrmaliger Existenz auf der Platte, gelöscht. Sinnvoll anwendbar sind alle DOS-residenten und -transienten Befehle, die sich auf ein Verzeichnis des Datenträgers beziehen. Befehle wie MODE oder DEBUG sind daher nicht unbedingt zu empfehlen. Der Rechner stürzt bei der Ausführung des Befehls zwar nicht ab, doch das Ergebnis ist eher unbefriedigend.

Befehle, die mit dem sogenannten Pipe-Zeichen (| = ALT – 142) zusammengesetzt sind, bedeuten für das Programm keine Probleme. Zum Beispiel bewirkt die Eingabe „dir |sort |more", eine sortierte, seitenweise Ausgabe des aktuellen Verzeichnisses. Dabei schreibt DOS das Verzeichnis in eine temporäre Datei, aus der sich der nachfolgende Befehl, in diesem Fall „sort", die Daten herausholt, bearbeitet und seinerseits in eine zweite temporäre Datei schreibt. Das nächste Kommando (more), holt sich dann wiederum die Daten aus dieser zweiten Datei und gibt sie anschließend seitenweise auf dem Bildschirm aus. Temporäre Dateien werden vom DOS zum Zwischenspeichern von Daten benutzt und bei einer eventuellen Ausgabe des gesamten Verzeichnisses zwar mit ausgegeben, nach der Bearbeitung aber wieder gelöscht.

Da „UeberAll" die DOS-Befehle nicht selbst interpretiert, sondern sie nur an den Kommando-Prozessor weitergibt, werden sie genauso abgearbeitet wie auf der DOS-Ebene. „UeberAll" wählt nach der Abarbeitung lediglich ein neues Verzeichnis aus und gibt den Befehl dann wiederholt an COMMAND.COM weiter.

Befehlsweitergabe

Das Programm selbst besteht aus drei Teilen: dem Hauptprogramm, der Prozedur FindDir und der Funktion OK. Dem Hauptprogramm stehen mit den Turbo-Pascal-Funktionen ParamCount und ParamStr die Anzahl und der Text der eingegebenen Parameter zur Verfügung, woraus es eine Kommandozeile baut. Sind keine Parameter angegeben, bricht das Programm seine Arbeit ab und weist mit einem Beispiel auf den Fehler hin. Andernfalls wird zur Sicherheit noch mal gefragt, ob der Befehl wirklich überall ausgeführt werden soll.

Die Tastaturabfrage übernimmt dabei die Funktion OK, die eingegebene Kleinbuchstaben zuerst mit der Funktion UpCase in Großbuchstaben umwandelt. Akzeptiert werden nur die Tasten „J", „Y" und „N". Im Falle einer negativen Antwort („N") liefert OK ein FALSE zum Hauptprogramm zurück, worauf das Programm dann abbricht. Bei einer positiven Antwort („J" oder „Y") liefert OK ein TRUE.

Weiter geht es im Hauptprogramm mit dem Befehl GetEnv, der feststellt, wo sich COMMAND.COM befindet, und den Suchpfad dann in der Variablen ExecPath ablegt. Um nach Beendigung des Programms wieder das Verzeichnis anzuwählen, das zur Zeit des Programmstarts aktuell war, muß dieses noch in der Variablen OrgPfad abgelegt werden. Danach startet die eigentliche Suchroutine FindDir.

Der erste Teil dieser Prozedur prüft, ob eine Taste gedrückt wurde. Falls ja, kann das Programm mit „J" abgebrochen oder mit „N" weitergeführt werden. Bevor es mit der Übergabe der Kommandozeile an COMMAND-.COM weitergeht, werden mit dem Befehl Swap Vectors, die Werte einiger Interrupt-Vektoren mit den Inhalten der globalen Variablen SaveInt getauscht. Das hat den Vorteil, daß das durch den angegebenen DOS-Befehl aufgerufene Programm mit den Original-DOS-Vektoren arbeiten kann. Bei einem eventuell auftretenden Fehler landet dieses Programm dann nicht in der Laufzeitbibliothek von Turbo-Pascal, sondern im DOS. Nach der Befehlsabarbeitung sollte SwapVectors erneut aufgerufen werden, um die Inhalte der Interrupt-Vektoren und der von SaveInt wieder zurückzutauschen.

Rekursive Suche

Der Befehl Exec reicht die Kommandozeile an COMMAND.COM weiter, der den DOS-Befehl dann im derzeit aktuellen Verzeichnis ausführt. Die Aufgabe des Programms besteht darin, festzustellen ob noch weitere Unterverzeichnisse existieren. Dazu sucht es mit FindFirst den ersten Eintrag im Verzeichnis, der anschließend näher untersucht wird.

```
1   (*$R+,S+,I+,B-,M 4096,0,0 *)  (* M <-- sehr wichtig! *)

2   Program UA;

3   (* Ein beliebiges Programm oder DOS-Befehl wird UeberAll,
4    d.h. im aktuellen Verzeichnis und allen Unterverzeichnissen ausgeführt.
5    Dr. Michael Bach / c't  1989 *)

6    Uses Crt, Dos;

7    Const
8     Version = 'Vs. 1.2, 17.7.89 MB';

9    Var
10     ExecPath, ComLine, OrgPfad  : PathStr;
11     i                           : Integer;


12    Function Ok: Boolean;

13    Var
14      c  : Char;
15      b  : Boolean;

16    Begin
17      Write(#7,'Ok? ');
18      Repeat
19        c:= UpCase(ReadKey);                        (* Kleinbuchstaben zu Großbuchstaben *)
20        b:= c in ['J','Y','N'];                     (* Prüfung auf gültige Eingabe *)
21        If not b then Write(#7);                    (* Wenn nicht, dann weiter warten *)
22      Until b;
23      ok:= c in ['J','Y'];                          (* ok ist TRUE, wenn J oder Y *)
24      WriteLn(c);
25    End;


26    Procedure FindDir;

27    Var
28      f  : SearchRec;

29    Begin
30      If KeyPressed then                            (* Wenn Taste gedrückt, dann *)
31      Begin
32        WriteLn;  Write('"UA" Abbrechen ');         (* fragen ob beendet werden soll *)
33        If ok then                                  (* Wenn ja, dann *)
34        Begin
35          ChDir(OrgPfad);  Halt;                    (* Start-Directory aufrufen *)
36        End;                                        (* und Programm beenden *)
37      End;
38      SwapVectors;  Exec(ExecPath, ComLine);  SwapVectors;
                                                      (* Vectoren tauschen und Parameter an *)
                                                      (* COMMAND.COM übergeben *)
39      FindFirst('*.*', Directory, f);               (* Ersten Eintrag im aktuellen *)
                                                      (* Directory suchen *)
40      While DosError = 0 do                         (* Wenn kein DOS-Fehler, dann *)
41      Begin                                         (* prüfen ob Eintrag ein Directory ist *)
42        if (f.attr = Directory) and (Pos('.', f.Name) = 0) then
43        Begin                                       (* Wenn ja, dann Directory wechseln *)
44          ChDir(f.Name);  FindDir;  ChDir('..');    (* Rekursiver Prozedur-Aufruf *)
45        End;
46        FindNext(f);                                (* Nächsten Eintrag suchen *)
47      End;
48    End;

49    Begin    (* Hauptprogramm *)
50      WriteLn;  WriteLn('*** UeberAll *** '+Version);
51      ComLine:= '';
52      For i:= 1 to ParamCount do ComLine:= ComLine+ ' '+ ParamStr(i);
53      (*  comline:='dir *.bak'; {zum Testen} *)
54      If ComLine = '' then                          (* Werden keine Parameter angegeben, *)
55      Begin                                         (* dann Text ausgeben *)
56        WriteLn('Parameter erwartet: "UA <aktion>". Dann wird <aktion>');
57        WriteLn('im aktuellen und allen Unterverzeichnissen ausgeführt.');
58        WriteLn('Beispiel: "UA dir *.bak"');
59        Halt;                                       (* und Programm abbrechen *)
60      End;
61      Write('Wirklich UeberAll "',ComLine+'" ');
62      If not ok then Halt;                          (* Ist ok = FALSE, dann Programm beenden *)
63      ExecPath:= GetEnv('COMSPEC');                 (* Suchweg zum COMMAND.COM herausfinden *)
64      ComLine:= ' /C'+ ComLine;
65      GetDir(0, OrgPfad);                           (* ermitteln des aktuellen Directory *)
66      FindDir;                                      (* Prozedur FindDir aufrufen *);
67      ChDir(OrgPfad);                               (* Zum alten Directory zurückkehren *)
68    End.
```

Das Turbo-Pascal-Programm führt jeden angegebenen DOS-Befehl im aktuellen und allen tieferliegenden Verzeichnissen aus.

Die beiden ersten Einträge eines Unterverzeichnisses sind zwar als Verzeichnisse ausgezeichnet, beginnen aber mit einem Punkt. Da sie aber nur Informationen über das aktuelle Verzeichnis enthalten, sind sie keine Verzeichnisse im eigentlichen Sinn und müssen daher übersprungen werden. Das Programm ruft ein Unterverzeichnis daher nur dann auf, wenn der gefundene Eintrag als Verzeichnis markiert und das erste Zeichen des Namens kein Punkt ist.

Findet das Programm ein solches Verzeichnis, so wählt es dieses als aktuelles Verzeichnis an und führt den gewünschten Befehl darin aus. Danach beginnt die rekursive Suche von neuem. Sind keine weiteren Unterverzeichnisse mehr vorhanden, wird mit CHDIR (..) ins übergeordnete Verzeichnis zurückgekehrt und dort geprüft, ob noch weitere Verzeichnisse vorhanden sind. Sind alle Verzeichnisse abgearbeitet, endet das Programm. (kp)

Nachdruck aus c't – Magazin für Computertechnik, Heft 10/1989

TERMINE

10. Internationale Fachtagung „Galvanotechnik 1990"

WER? Kammer der Technik, Fachverband Elektrotechnik und Bezirksverband Karl-Marx-Stadt

WANN? 9. bis 11. Mai 1990

WO? Stadthalle Karl-Marx-Stadt

WAS?
- komplexe Prozesse in der Galvanotechnik, insbesondere Beschichtungsverfahren, Stoffkreisläufe, Wertstoffrecycling, rechnergestützte Prozeßplanung und -realisierung
- galvanische Verfahren für die Mikroelektronik

WIE? Einladungen fordern Sie bitte an bei: Kammer der Technik, Bezirksvorstand Karl-Marx-Stadt, Annaberger Straße 24, Karl-Marx-Stadt, 9001; Tel. 6 21 41

13. Internationales Seminar über Datenbankbetriebssysteme

WER? Forschungsinstitut für Computer und Informatik (ITCI) und Territoriales Rechenzentrum Constanța (CTCE)

WANN? 17. bis 22. September 1990

WO? Mamaia, Rumänien

WAS?
- Datenbanktheorie, Modelle und Werkzeuge
- Datenbank-Entwurfsmethodologien und -Sprachen
- objektorientierte, deduktive und temporale Datenbanken
- Schnittstellen für natürliche Sprachen
- Datenbank-Maschinen

WIE? Die Veranstaltungssprache ist Englisch. Nähere Informationen erhalten Sie über: VEB Leitzentrum für Anwendungsforschung Berlin, (Dr. Lange oder Dr. Oßwald), Jacques-Duclos-Straße 47/52, Berlin, 1156.

Schreibschutz per Software

Dr. Christian Hanisch, Berlin

Es gibt Anwendungsfälle, bei denen die Festplatte des PCs zugänglich sein muß, aber Schreibzugriffe – gewollte und ungewollte (beispielsweise bei Virenbefall) – nicht zugelassen werden sollen.

Mit dem Programm HDONLY (Hard Disk ONLY read), das als TSR-Programm (Terminate and Stay Resident) den INT-13H-Handler des Betriebssystems durch eine entsprechende eigene Routine ersetzt, ist es möglich, einen Schreibschutz per Software zu garantieren. Durch den eigenen INT-13H-Handler werden Schreibzugriffe entweder auf die gesamte Festplatte oder auf eines der Laufwerke A und B (Diskette) bzw. C, D und E (Festplatte) abgewiesen. Der Zugriff auf die RAM-Disk ist davon nicht betroffen. Bei einer Aufteilung der Festplatte in mehrere logische Festplattenlaufwerke (HDONLY nimmt maximal drei logische Laufwerke C, D und E an) ist es zweckmäßig, das Laufwerk C als sogenanntes SYSTEM-Laufwerk *Read Only* zu setzen, um dadurch den Bestand an Standardsoftware vor Modifikationen zu schützen.

Das Problem des *versehentlichen* Formatierens der Festplatte wird durch Angabe des Parameters **F** oder **f** beim Aufruf von HDONLY gelöst. Es sind zwar Lese- und Schreibzugriffe auf die Festplatte erlaubt, ein Formatieren von Teilen oder der gesamten Festplatte aber wird zurückgewiesen:

HDONLY f<CR> Kein Formatieren erlaubt!

Der Aufruf ohne Parameter sperrt die gesamte Festplatte gegen Schreiben und ist beispielsweise bei einer diskettenorientierten Arbeitsweise auf einem PC mit sachgebietsorientiert eingerichteter Festplatte zu empfehlen. Die Nutzer können dabei auf die Standardsoftware lesend zugreifen – haben aber die Festplatte sonst nicht zur Verfügung.

Darüber hinaus könnten durch weitere Schutzmaßnahmen (Password, SaveDir) gewisse Dateien auch gegen einen Lesezugriff abgesichert werden. Da HDONLY auch die Diskettenlaufwerke A und B mit Software-Schreibschutz versehen kann, ist eine bequeme Möglichkeit gegeben, ohne die Diskette *abzukleben*, gewisse Tests zur Virensuche auf Disketten vorzunehmen. Allerdings hilft das nur gegen Viren, die sich mit diesem INT-13H-Trick *fangen* lassen.

Auch beim Kopieren zur Datensicherung vom Laufwerk A nach B kann es nützlich sein, das Laufwerk A *Read Only* zu setzen, um nicht irrtümlich die alten Backup-Dateien vom Laufwerk B auf die aktuellen Dateien des Laufwerks A zu *sichern*. Das speicherresidente Programm HDONLY ist natürlich nur ein Softwareschutz und hat damit seine entsprechenden Grenzen. Es könnte in der AUTOEXEC.BAT als letztes der resident zu ladenden Programme aufgerufen werden. Dadurch kann es bei Bedarf mit einem (residenten) Programm zum Speichermanagement (beispielsweise mit dem *vorher* geladenen Ram Free) wieder bequem entfernt werden. Nach der Entfernung von HDONLY aus dem RAM ist der Schreibschutz wieder unwirksam. Dieses Vorgehen ist bei Diskettenarbeit deutlich schneller als Diskette abkleben und danach den Schreibschutz *physisch* wieder entfernen.

Es gibt Softwarewerkzeuge und einzelne Programme, die während ihrer Abarbeitung gewisse Parametereinstellungen auf die Platte zurückschreiben – beispielsweise die integrierten Pakete. In diesen Fällen muß über Batch-Dateien erst einmal eine Kopie auf die RAM-Disk oder ein ungeschütztes Festplattenlaufwerk gebracht und die eigentliche Abarbeitung dann von dort gestartet werden.

Bild 1 zeigt den Assembler-Quelltext des Programms **HDONLY.ASM**, der so abgetippt und mit **MASM, LINK** und **EXE2BIN** in die Datei **HDONLY.COM** überführt werden kann. Zum weitgehend selbsterklärenden Quelltext einige Bemerkungen:

- Das Programm HDONLY.ASM besteht als TSR-Programm aus zwei Teilen, dem eigentlichen *neuen* INT-13H-Handler (ab Marke MY_INT13 bzw. TSR_Begin) und dem Initialisierungsteil (ab Marke INIT_VECTORS bis zum Programmende).
- Im Initialisierungsteil wird die Auswertung der ParamStr-Information vorgenommen und ein Test durchgeführt, der ein wiederholtes Laden des Programms verhindert. Letzteres geschieht mit Hilfe des Makros XREADY. Dabei wird der Code des neu zu installierenden Programms mit dem Code aller bereits

```
  1:     page 88,132
  2:     title    HDONLY.ASM    -   Schreibschutz per Software
  3: ; Version 2.0
  4: COMMENT @
  5:   Das Programm faengt ueber einen "neuen" INT 13 -Handler
  6:   alle Schreibbefehle auf:
  7:  1) die Harddisklaufwerke (80h .. 82h)
  8:     - ausser RAM-Disk´s -
  9:  2) das beim Aufruf als <ParamStr>-Information angegebene
 10:      Laufwerk
 11:     - C..E (Festplatten) und  A..B (Disketten) -
 12:  3) - ein FORMATieren der Festplatte, falls als <ParamStr>-
 13:       Information "F" angegeben wird,
 14:  ab.
 15:                    (C)  Ch. Hanisch
 16: @                                                    23.06.89
 17:
 18: ;     M  a  c  r  o  s:
 19: DOS  macro func,parm
 20:      IFNB <parm>
 21:      mov  ax,func*256+parm
 22:      ELSE
 23:      mov  ah,func
 24:      ENDIF
 25:      int 21h
 26:      endm
 27:
 28: LINK  macro int,oldintloc,newintloc
 29:       mov al,int
 30:       DOS 35h
 31:       mov Word PTR oldintloc,bx
 32:       mov Word PTR oldintloc+2,es
 33:       mov dx,OFFSET newintloc
 34:       mov al,int
 35:       DOS 25h
 36:       endm
 37:
 38: KEEP  macro initbegin,startpgm
 39:       mov  ax,3100h    ; T)erminate and S)tay  R)esident
 40:       mov dx,((OFFSET initbegin-OFFSET startpgm)+10Fh) SHR 4
 41:       int 21h
 42:       endm
 43:
 44: XREADY  macro CmpStart,CmpEnd,SchonDaMarke
 45:          LOCAL NextMCB,NoMCB
 46: ; Die undokumentierte Funktion 52h liefert einen Zeiger auf
 47: ; eine Liste, an deren Anfang-2 der ZEIGER auf den ersten MCB.
 48: ; (Memory Control Block) steht.
 49:         mov ah,52h
 50:         int 21h
 51:         mov dx,es:[bx-2]
 52:         mov bx,cs
 53:         mov es,bx
 54:         push ds
 55: ; Hat der aktuelle MCB die Kennung: "M"
 56: ; dann ist er belegt.
 57: NextMCB:
 58:         mov ds,dx
 59:         xor si,si
 60:         lodsb
 61:         cmp al,"M"
 62: ; Falls nicht "M", dann ist die Liste beendet;
 63: ; das Programm ist noch nicht resident geladen.
 64:         jne NoMCB
 65: ; An OFFSET 1 steht die Segmentadresse des PSP
 66: ; (P)rogram S)egment P)refix - Programmvorspann).
 67:         lodsw
 68:         push ax
 69: ; An OFFSET 3 steht die Länge des momentanen MCB
 70: ; in Paragraphen - 1.
 71:         lodsw
 72:         inc ax
 73:         add dx,ax
 74: ; Wenn der gerade untersuchte MCB der des aktuellen
 75: ; Codesegments ist, dann wird die Prüfroutine übersprungen.
 76:         pop ax
 77:         cmp ax,bx
 78:         je NextMCB
 79: ; Jetzt werden die beiden Programmteile miteinander
 80: ; verglichen:
 81:         mov ds,ax
 82:         mov si,OFFSET CmpStart
 83:         mov di,si
 84:         mov cx,OFFSET CmpEnd - OFFSET CmpStart
 85:         REPE Cmpsb
 86:         jne NextMCB
 87: NoMCB:
 88:         pop ds
 89:         jne NoSchonX
 90:         jmp SchonDaMarke
 91: NoSchonX   equ $
 92:         endm
 93:
 94: ; ---------------------------------------------------------------
 95:
 96: code SEGMENT  ´code´
 97:         ASSUME cs:code,ds:nothing,es:nothing
 98:         org 100h
 99: TSR_Begin: jmp INIT_VECTORS ; Initialize and Attach to DOS
100: HDONLY PROC far
101: LWTyp  db 02h,02h        ; Lw. A und B Diskettenlaufwerke
102:        db 24 DUP (0FFh)
103: ;          Laufwerkstypen in A..B / C/D/E..Z
104: ;  FFh  -  RAM-Disk oder ungültiges Laufwerk
105: ;  05h  -  Festplattenlaufwerk
106: ;  02h  -  Diskettenlaufwerk (720 KByte)
107: LWArt  db 10000000B      ; Maske fuer Art des Laufwerks
108: OLD_INTLOC label DWORD  ; Address for old INT 13-Routine
109:        dd ?
110: ; ==============================================================
111: MY_INT13 PROC NEAR
112:           pushf
113:           test dl,cs:LWArt                  ; Maske abtesten
114:           jz   Do_It_1
115: CallGo:   call DSKTYP
116:           jc  Do_It_2     ; CF gesetzt --> RAM-Disk oder
117:                           ;                 ungueltiges Lw.
118:           cmp  ah,3       ; Funktion: Sector schreiben       ?
119: F1:       je   Error
120:           cmp  ah,0Bh     ; Funktion: Sector schreiben       ?
```

```
121: F2:      je   Error
122:          cmp  ah,0Fh      ; Funktion: Sectorbuffer schreiben ?
123: F3:      je   Error
124:          cmp  ah,7        ; Funktion: Ab Spur N formatieren  ?
125:          je   Error
126:          cmp  ah,6        ; Funktion: Spur formatieren       ?
127:          je   Error
128:          cmp  ah,5        ; Funktion: Sector formatieren     ?
129:          jne  Do_It_2
130: Error:
131:          mov  ah,0BBh             ; Fehlercode und CF setzen
132:          popf
133:          stc                      ; DOS ruft dann INT 24h
134:          ret 2                    ;  HDONLY beenden !!
135:
136: Do_It_1:
137:          or   Byte PTR cs:LWArt,00000000B
138:          jz   CallGo              ; Bei Diskettenlaufwerk A:
139: Do_It_2: popf
140:          jmp  DWord PTR cs:OLD_INTLOC  ;Rufe "alten" INT 13
141: ; ---------------------------------------------------------
142: DSKTYP   PROC  near
143: ; RAM-Disk-Laufwerk bzw. ungültiges Laufwerk herausfiltern
144: ; INPUT: DL - Laufwerk: 00h..01h..7Fh  (Diskette);
145: ;                       80h..83h..FFh  (Festplatte/RAM-Disk)
146: ;OUTPUT: CF gesetzt - RAM-Disk oder ungueltiges Lw.
147:          push ax
148:          push si
149:          xor  ax,ax
150:          mov  al,dl
151:          and  al,10000000B  ; Kein Festplattenlaufwerk ?
152:          cmp  al,10000000B
153:          jne  NoRAMDx
154:          mov  al,dl
155:          sub  al,126         ; 80h --> 02h; 81h --> 03h; ...
156:          mov  si,ax
157:          cmp  cs:LWTyp[si],0FFh        ; RAM-Disk-Laufwerk,
158:          jne  NoRAMDx
159:          stc
160:          jmp  Short Retur
161: NoRAMDx: sahf
162: Retur:   pop   si
163:          pop   ax
164:          ret
165: DSKTYP   endp
166: MY_INT13 endp
167: TSR_End  equ    $
168: HDONLY   endp
169: ; ---------------------------------------------------------
170: ;  I n i t i a l i s i e r u n g :
171: ; ---------------------------------------------------------
172:               ASSUME ds:code,es:nothing
173: RAMDSK   PROC near
174: ; INPUT: DL - Laufwerk: 01h..02h (Diskette);
175: ;                       03h..05h (Festplatte/RAM-Disk)
176: ;OUTPUT: LWTest - Typ des Laufwerks: 05h (Festplatte);
177: ;                                    02h (Diskette);
178: ;                                    FFh (Ungültiges Lw.)
179: ;              CF gesetzt - RAM-Disk oder ungueltig
180:          push bx
181:          push cx
182:          push dx
183:          push ax
184:          mov  bl,dl          ; Laufwerk: 01h..05h
185:          mov  ah,44h
186:          mov  al,0Dh
187:          mov  ch,8
188:          mov  cl,60h
189:          mov  dx,OFFSET ds:DPB
190:          int  21h
191: ;        Bei RAM-Disk oder ungueltigem Lw.
192: ;        wird CF gesetzt !!
193:          push si
194:          mov  si,dx
195:          mov  al,Byte PTR ds:[si+1]
196:          mov  Byte PTR cs:LWTest,al
197:          pop  si
198:          pop  ax
199:          pop  dx
200:          pop  cx
201:          pop  bx
202:          ret
203: RAMDSK   endp
204: ; >>>>>>>>>>>>>>>>>> Hier geht's los !!!! <<<<<<<<<<<<<<<<<<
205: INIT_VECTORS PROC near
206:       push  cs            ; Initialisieren des Daten-Segments
207:       pop   ds
208:       xor   dx,dx            ; DX definiert setzen auf Null
209:       xor   ax,ax            ; AX definiert setzen auf Null
210:       mov   es,ax            ; ES definiert setzen auf Null
211:       mov   si,80h
212:       lodsb
213:       add   si,ax
214:       mov   Byte PTR ds:[si],00h
215:       XREADY MY_INT13,TSR_End,SchonDa
216: ; <ParamStr>-Information aus der Aufrufzeile:
217: ; Aufruf:
218: ; =======     HDONLY [<Laufwerk>]<CR>
219: ;
220: ;                      <Laufwerk>::= A..B  und C..E oder
221: ;                                   F)ORMATieren gesperrt.
222: ; Mit der <ParamStr>-Information "F" bzw. "f" wird die
223: ; Festplatte nur für FORMATieren gesperrt, sonstige
224: ; Schreibzugriffe sind erlaubt.
225:    mov  si,81h
226: Leer:
227:      lodsb
228:      cmp   al,0
229:      je    OhneParm
230:      cmp   al,20h            ; <SP> (Blank)   ?
231:      je    Leer
232:      cmp   al,09h            ; <TAB> (^I)     ?
233:      je    Leer
234:      cmp   al,"/"            ; Parm-Separator ?
235:      je    Leer
236:      mov   dl,al
237:      cmp   dl,"a"            ; Umwandeln in GROSS-Buchstaben
238:         jl     HDok
239:         cmp    dl,"z"
240:         jg     HDok
241:         sub    dl,20h              ; oder: and dl,255-32
242: HDok:
243:         cmp    dl,"A"              ; Diskettenlaufwerke A und B
244:         jl     OhneParm
245:         cmp    dl,"E"              ; Explizit nur bis Laufwerk E
246:         jg     OhneParm
247:         mov    LW,dl               ; Laufwerksname sichern f. Anz.
248:         sub    dl,64               ; Convert to Binary; Lw. A --> 01h
249:         mov    LWTest,0FFh         ; Wird von RAMDSK dann gesetzt !
250:         call   RAMDSK
251:         jc     OhneParm
252:         cmp    LWTest,05h          ; Festplattenlaufwerk ?
253:         jne    HDNo
254:         add    dl,125              ; 03h --> 80h; 04h -> 81h; ...
255:         mov    LWArt,dl            ; Neue Maske fuer Lw. C/D/E
256:         jmp    short OhneParm
257: LWTest         db  0FFh            ; Zwischenspeicher fuer RAMDSK
258: HDNo: dec  dl
259:         mov  LWArt,dl              ; Neue Maske fuer Lw. A/B
260: ; <ParamStr>-Information ::= F  ====> no FORMATieren der
261: ;                                          Festplatte
262: OhneParm:
263:         cmp     dl,"F"
264:         jne     attach
265:         mov     ax,9090h           ; Zwei NOP-Befehle
266:         mov     Word PTR F1,ax
267:         mov     Word PTR F2,ax
268:         mov     Word PTR F3,ax
269: attach:
270: ;       Laufwerkstypen feststellen  --> LWTyp[i]
271:         mov     si,3             ; Laufwerk C .. Z
272:         mov     di,2
273:         mov     cx,24
274: Zyklus:
275:         mov     dx,si
276:         mov     Byte PTR LWTyp[di],0FFh ; RAM-Disk-Laufwerk ?
277:         call    RAMDSK
278:         jc      RAMD_Ist
279:         mov     al,LWTest
280:         mov     Byte PTR LWTyp[di],al
281: RAMD_Ist:
282:         inc     si
283:         inc     di
284:         loop    Zyklus
285:         LINK    13h,OLD_INTLOC,MY_INT13
286:         mov     dx,OFFSET message   ; Installations-Meldung
287:         mov     ah,9
288:         int     21h
289: ; ---------------------------------------------------------
290:         KEEP    TSR_End,TSR_Begin
291: ; ---------------------------------------------------------
292: SchonDa:
293:      mov dx,OFFSET Errormessage ; Fehlermeldung
294:      mov ah,9
295:      int 21h
296:      mov ax,4C80h              ; errorlevel 128
297:      int 21h                   ; Return  to DOS
298: ; ---------------------------------------------------------


299: message db 13,10,27,'[K',13,10,27,'[K'
300:  db 13,10,27,'[K'
301:  db        '╔═══════════════════════════════════╗'
302:  db 13,10,'║    ',27,'[7mH',27,'[m)ard-',27,'[7mD'
303:  db 27,'[m)isk ',27,'[7mONLY',27,'[m read     ( INT 13 ) ║'
304:  db 13,10,'║                                   ║'
305:  db 13,10,'║ Aufruf:                           ║'
306:  db 13,10,'║         HDONLY [<Laufwerk>]<CR>   ║'
307:  db 13,10,'║                                   ║'
308:  db 13,10,'║ <Laufwerk>::= A..B und C..E -'
309:  db 27,'[7m F)ORMAT no',27,'[m║'
310:  db 13,10,"║              ::= F  -  F)ormatieren der  ║"
311:  db 13,10,"║                         Harddisk verboten !  ║"
312:  db 13,10,'║                                   ║'
313:  db 13,10,'║ Schreibschutz per S O F T-WARE für die ║'
314:  db 13,10,'║ gesamte Festplatte bzw. das beim Auf-  ║'
315:  db 13,10,'║ ruf angegebene <Laufwerk>: '
316: LW  db 20h
317:  db        '║'
318:  db 13,10,"║                                   ║"
319:  db 13,10,'║ HDONLY schützt vor "Viren-Infektionen" ║'
320:  db 13,10,'║ und ermöglicht Laufwerk C als Read Only║'
321:  db 13,10,'║ SYSTEM-Laufwerk.                  ║'
322:  db 13,10,'╚ 2.0 ═════════ (C) Ch. Hanisch ═══╝'
323:  db 13,10,27,'[K',13,10,27,'[K',24h
324: Errormessage db  13,10,27,'[K',13,10,27,'[K'
325:  db 13,10,'╔═══════════════════════════════╗'
326:  db 13,10,'║                               ║'
327:  db 27,'[7m H D O N L Y ',27,'[m         ║'
328:  db 13,10,'║         on INT 13             ║'
329:  db 13,10,'║    already resident !!        ║'
330:  db 13,10,'║                               ║'
331:  db 13,10,'║ Schreibschutz per S O F T W A R E║'
332:  db 13,10,'║  schützt vor "Viren-Infektionen" ║'
333:  db 13,10,'║                               ║'
334:  db 13,10,'╚ 2.0 ══════════ (C) Ch. Hanisch ╝'
335:  db 13,10,27,'[K',13,10,27,'[K',24h
336: DPB equ Errormessage            ; DPB  db  50 DUP (88)
337: INIT_VECTORS ENDP
338: code ENDS
339:       end TSR_Begin
```

Bild 1 Quelltext des Assemblerprogramms HDONLY.ASM

resident geladenen verglichen. Wird in einem Fall Gleichheit festgestellt, so ist das Programm bereits im Speicher installiert; der Nutzer wird darüber informiert.

- In der Prozedur RAMDSK wird geprüft, ob die in dem Feld LWTyp aufzuführenden Laufwerke von A bis Z Festplatten (05H), Disketten (02H) oder ungültig bzw. RAM-Disks (0FFH) sind. Die Informationen aus dem Feld LWTyp werden später von der Prozedur DSKTYP beim Aufruf des INT-13H ausgewertet.
- Mit den Makros LINK und KEEP wird die TSR-Routine des *neuen* INT-13H-Handlers resident. Eine entsprechende Nachricht (Message bzw. Errormessage) wird zur Information des Bedieners ausgegeben, die aber durch:

HDONLY c >nul

in BATCH-Dateien ins Nichts (nul) umgeleitet werden kann.

- In der Prozedur MY_INT13 werden mittels einer Maske LWArt, die im Initialisierungsteil entsprechend der in ParamStr übergebenen Laufwerksangabe gestellt wurde, die INT-13H-Anforderungen überprüft. Im Register AH werden die Codes für die entsprechenden Schreiboperationen übergeben. Falls ein passender Code festgestellt wird, verzweigt das Programm zur Marke Error und setzt einen Fehlercode, der zum Aufruf des INT 24H führt.

Das Betriebssystem schreibt eine Fehlermeldung, die vom INT-24H ausgelöst wird. Der Bediener muß mit *Abbruch* antworten, womit der Schreibversuch beendet wird. Falls kein passender Code in AH festgestellt wird, geht die Steuerung zum *originalen* INT-13H zur Ausführung der Anforderung.

Im Falle der Sperre nur für das Formatieren werden an den Marken F1 bis F3 drei NOP-Befehle eingefügt, so daß die Auswertungen der Prüfungen auf die entsprechenden Codes zum *Sector schreiben* usw. entfallen.

Literatur

/1/ Smode, D.: MS-DOS Arbeitsbuch. München: Francis Verlag GmbH 1987

/2/ Smode, D.: MS-DOS für Insider. München: Francis Verlag GmbH 1987

/3/ MS-DOS Technical Reference Encyclopedia. – Microsoft Press 1986

/4/ Sternberg, G.: Harddisk softgeschützt. München: Francis Verlag GmbH, mc (1989) 4, S. 103

Prozedurrufe zurückverfolgen

Olaf Ginter, Bad Klosterlausnitz

Mit Hilfe vorgefertigter Programmteile (Bibliotheksroutinen) ist auch mit Turbo-Pascal die Programmerstellung effektiver möglich. Allerdings können solche Programmbausteine nicht völlig unabhängig von globalen Bedingungen eingefügt werden. Falsche Parameterfestlegungen, die bei der Programmierung der *Bausteine* nicht vorausbedacht wurden, können dann Programmfehler und -abbrüche verursachen. Wird als Abbruchstelle ein solcher vorgefertigter Programmbaustein ermittelt, dann entsteht die Frage, welcher übergeordnete Programmteil diesen benutzt hat und dabei den Fehler verursachte. Ab Version 5.0 liefert der in die Entwicklungsumgebung integrierte Debugger über die Tastenkombination [Ctrl]-F3 („Call Stack") und das Setzen von Unterbrechungspunkten Hilfsmittel zur Klärung dieser Fragestellung. Wie aber kann man sich bei den älteren Compilerversionen helfen?

Beispielsweise kann über Monitorausschriften die Prozedurruffolge ermittelt werden. Eleganter ist es aber, nicht alle Aufrufstellen zu suchen und mit Monitorausschriften zu erweitern, sondern einfach im Fehlerfall über den Prozessor-Stack die Aufrufe zurück zu verfolgen.

Wie das bei den Compiler-Versionen 3.x für 16-Bit-Rechner erreichbar ist, wurde in /1/ gezeigt. Im Bild 1 wird Ihnen nun eine Include-Datei zur Fehlersuche für die 8-Bit-Version vorgestellt.

Der Compiler oder die Entwicklungsumgebung, in die er integriert ist, erlaubt unter den Optionen auch die Suche nach einer bestimmten Programmstelle deren Adresse hexadezimal eingegeben wird. Bei erfolgreicher Suche wird in den integrierten Editor verzweigt und der Kursor auf die entsprechende Quelltextstelle gebracht. Eine Besonderheit ist dabei, daß die eingegebene Suchadresse relativ zur Startadresse des Programms, also als Offset, interpretiert wird. Auf Grund der Besonderheiten des Compilers befinden sich nach Eintritt in eine gerufene Prozedur (zur Ausführungszeit der ersten Anweisung) keine Parameter mehr und auch keine lokalen Variablen auf dem Prozessor-Stack, sondern lediglich die Return-Adresse. Diese kann man sich nun durch *Inline* von Assemblercode oder, wie hier gezeigt, über den StackPtr holen, davon die Startadresse des Programms subtrahieren und das Ergebnis hexadezimal ausgeben. Dies erledigt die Prozedur **TraceBack** von der Stelle an, wo sie selbst aufgerufen wurde (also innerhalb der zu untersuchenden Unterprozedur). Vorher muß festgelegt werden, bis zu welcher Programmebene zurückverfolgt werden soll. Das geschieht durch Aufruf der Prozedur **InitBackTracer** auf der entsprechenden Programmebene, also beispielsweise gleich zu Beginn des Hauptprogramms, wie das auch im Demonstrationsprogramm (Bild 2) gezeigt wird. So kann jeder Aufruf zurück bis zur Hauptprogrammebene verfolgt werden. Nach den von **TraceBack** ausgegebenen Adressen kann dann der Compiler suchen.

Da die Startadresse des Programms von der Compilerversion, dem Compilermodus und ihrer expliziten Einstellung durch den Nutzer abhängt, ist die Konstante **StartAdr** im Bild 1 entsprechend anzupassen. Der Wert ist dem Optionen-Menü oder der Compiler-Ausschrift zu entnehmen.

```
 1: {-----------------------------------------------------------------
 2: Rueckverfolgung von Aufrufen einer Unterprozedur in Pascal-T(880/S):
 3:   Zu Beginn des Hauptprogramms wird "InitBackTracer" aufgerufen und
 4: in der interessierenden Unterprozedur dann "BackTrace". Diese Proze-
 5: dur gibt auf dem Bildschirm die Adressen der Programmstellen aus,
 6: von denen aus die untersuchte Unterprozedur gerufen wurde. Diese
 7: Adressen koennen dann im Optionen-Menue des Compilers mit der Funk-
 8: tion Fehlersuche gesucht werden.
 9: -----------------------------------------------------------------}
10: CONST StartAdr=¤20E2;      {Startadresse fuer Compilermodus Com-File}
11:
12: VAR   StartLevel:integer;
13:
14: PROCEDURE InitBackTracer;
15: BEGIN
16:   StartLevel:=StackPtr;
17: END;
18:
19: PROCEDURE TraceBack;
20: VAR ad:^integer;
21:
22: TYPE Hexzahl = STRING[4];
23:
24: FUNCTION Hexbyte(x:integer):Hexzahl;
25:  CONST hexziffer:ARRAY[0..15] OF char='0123456789ABCDEF';
26: BEGIN
27:   Hexbyte:=Hexziffer[(lo(x)SHR 4)]+Hexziffer[(lo(x)AND 15)];
28: END;
29:
30: FUNCTION Hexword(x:integer):Hexzahl;
31: BEGIN
32:    Hexword:=hexbyte(hi(x))+hexbyte(lo(x));
33: END;
34:
35: BEGIN
36:   writeln;
37:   writeln('======== Rueckverfolgung von Prozedur-Rufen =========');
38:   ad:=ptr(StackPtr);
39:   REPEAT
40:     ad:=ptr(ord(ad)+2);
41:     writeln(hexword(ad^-StartAdr-1));
42:   UNTIL ord(ad)>=StartLevel;
43: END;
```

***Bild 1** Listing der Include-Datei zur Rückverfolgung von Prozedurrufen*

```
 1: PROGRAM TraceBackDemo;
 2:
 3: {¤I TRACEBK.INC}
 4:
 5: PROCEDURE dummy1;
 6: BEGIN
 7:   traceback;
 8: END;
 9:
10: PROCEDURE dummy2;
11: BEGIN
12:   dummy1;
13: END;
14:
15: BEGIN
16:   initbacktracer;
17:   dummy2;
18: END.
```

***Bild 2** Demonstration einer Rückverfolgung von Prozedurrufen*

Literatur

/1/ Gieselmann, K.: Turbo Pascal auf der Spur. Pascal, Eschwege 3 (1988) 4, S. 80.

Computer-Club

Strukturiert programmieren in Basic

Dietmar Gralla, Berlin

Nicht allen Programmierern ist es möglich, eine Programmiersprache, die die strukturierte Programmierung unterstützt, auf ihren Rechnern einzusetzen. Es ist aber eine Forderung unserer Zeit, daß das statische Layout auch für andere Programmierer in kürzester Zeit lesbar ist. Deshalb soll in diesem Beitrag für die genannte Gruppe von Programmierern ein Weg gezeigt werden, wie auch Basic-Programmierer strukturiert programmieren können, obwohl dies in Basic nicht ohne weiteres möglich ist.

Das Variablenkonzept

Die Variablen werden als Steuerregister deklariert. Diese werden als SRn bezeichnet und auf 0 gesetzt. Soll eine Aktion ausgeführt werden, wird SRn auf einen von 0 verschiedenen Wert gesetzt.

Beispiel:
SRS ... Steuerregister Zeichensatz
SRS=0 ... Ausgangszustand
SRS=1 ... deutscher Zeichensatz
SRS=2 ... amerikanischer Zeichensatz

Vor Eintritt in die entsprechende WHILE-Schleife wird der Wert von SRn an SRnW übergeben. SRnW ist die für die WHILE-Schleife relevante Variable und kann verändert werden. Ist wie im Beispielprogramm eine einmalige Abarbeitung gefordert, wird vor Verlassen der Schleife SRnW ohne Bedingung auf einen Wert gesetzt, der dem Typ von SRnW, aber nicht der Eingangsbedingung entspricht.

Beispiel:

```
WHILE SRSW=1
   .
   .
   .
   SRSW=0
WEND
```

Dadurch erfolgt das Verlassen der Schleife, ohne daß das Steuerregister SRn verändert wurde, und es steht SRn für die nächste Abarbeitung bereit.

Der Programmaufbau

Für eine klare Strukturierung ist es notwendig, Sprunganweisungen (GOTO-Anweisungen), die aus der Zeile herausführen, möglichst zu umgehen. Dazu werden die IF-Anweisungen durch WHILE-Anweisungen ersetzt. Innerhalb der WHILE-Schleife wird eine Aktion in Abhängigkeit von einer Eingangsbedingung ausgeführt. Nach ausgeführter Aktion innerhalb der Schleife muß die Eingangsbedingung negiert werden, um eine weitere Abarbeitung zu unterdrücken.

Das Programm DEMO geht davon aus, daß dem Leser ein Bildschirm mit 24 Zeilen, 80 Spalten (BAB 2), ein Bildwiederholspeicher ab Adresse 63488d (F800H) und ein Drucker LX 86 zur Verfügung steht.
Auf das Grundmenü wird ein Hilfsmenü *Druckersteuerung gepoket*, das zuvor aus den DATA-Anweisungen in das Feld WF() geladen wurde. Der ursprüngliche Inhalt des Bildwiederholspeichers befindet sich anschließend im Feld WF(), wodurch die ursprüngliche Bildschirmausschrift wieder hergestellt werden kann und das Hilfsmenü wiederum im Feld WF() zur Verfügung steht (Unterprogramm 1240). Es wird an einem einfachen Beispiel gezeigt, wie der Inhalt des Bildwiederholspeichers gewechselt werden kann, ohne daß der Programmierer den ursprünglichen Zustand aufwendig im voraus erkennen und programmieren muß.
Der dem Hilfsmenü entsprechende Inhalt des Bildwiederholspeichers wurde mit dem Programm WINDOW im ASCII-Code ausgedruckt und kann sowohl aus den DATA-Anweisungen im Programm DEMO als auch aus einer Datei von Diskette gelesen werden.

Eine Erhöhung der Effizienz kann erreicht werden, wenn die Steuerregister mit DEFSTR S als STRING-Variablen deklariert werden. Dann kann die Kodierung für den deutschen Zeichensatz wie folgt aussehen:
SRS="d" oder wenn auf DEFSTR S nicht zurückgegriffen wurde: SRS$="d". Diese Varianten führen zu einer *verbalen Programmierung*.

Durch die gezeigte Technik erhöht sich die Lesbarkeit des Programms; sie erleichtert die Fehlersuche und verringert die GOTO-Anweisungen auf ein Minimum.

```
10 'WINDOW.BAS
20 DEF FNPK$(L,C)=CHR$(27)+CHR$(127+L)+CHR$(127+C)
30 ? CHR$(12)
40 DEFSTR Z
50 FOR K1=4 TO 20
60    ON K1-3 GOSUB 4000,5000,6000,7000,8000,9000,10000,11000,12000,13000,
                    14000,15000,16000,17000,18000,19000,20000
70    ? FNPK$(K1,12);ZEILE
80 NEXT K1
90 :
100 FOR K1=4 TO 20
110    FOR K2=12 TO 42
120       SPEICHER=PEEK(63488+80*(K1-1)+(K2-1))
130       LPRINT CHR$(SPEICHER);
140    NEXT K2
150 LPRINT
160 NEXT K1:LPRINT
170 IF TA$<>"W" AND TA$<>"w" THEN GOTO 170
180 FOR K1=4 TO 20
190    FOR K2=12 TO 42
200       SPEICHER=PEEK(63488+80*(K1-1)+(K2-1))
210       LPRINT USING"####";SPEICHER;
220    NEXT K2
230 LPRINT:NEXT K1
240 END
4000 ZEILE="###############################":RETURN
5000 ZEILE="#                             #":RETURN
6000 ZEILE="# Druckersteuerung            #":RETURN
7000 ZEILE="#                             #":RETURN
8000 ZEILE="# *Zeichensatz >d<eutsch      #":RETURN
9000 ZEILE="#               >a<merikanisch #":RETURN
10000 ZEILE="#                            #":RETURN
11000 ZEILE="# *Schriftart  >n<ormal      #":RETURN
12000 ZEILE="#              >k<omprimiert #":RETURN
13000 ZEILE="#                            #":RETURN
14000 ZEILE="# *Drucker     >r<uecksetzen #":RETURN
15000 ZEILE="#                            #":RETURN
16000 ZEILE="# *            >h<aenden     #":RETURN
17000 ZEILE="#                            #":RETURN
18000 ZEILE="# --->gewaehlt:> <           #":RETURN
19000 ZEILE="#                            #":RETURN
20000 ZEILE="##############################":RETURN
```

```
10 'DEMO.BAS
20 DATA 35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,
        35,35,35,35,35,35,35,35
30 DATA 35,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,
        32,32,32,32,32,32,32,35
40 DATA 35,32,68,114,117,99,107,101,114,115,116,101,117,101,114,117,110,103,32,
        32,32,32,32,32,32,32,32,32,32,35
50 DATA 35,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,
        32,32,32,32,32,32,32,35
60 DATA 35,32,42,90,101,105,99,104,101,110,115,97,116,122,32,62,100,60,101,117,
        116,115,99,104,32,32,32,32,32,32,35
70 :
80 DATA 35,32,32,32,32,32,32,32,32,32,32,32,32,32,32,62,97,60,109,101,114,105,
        107,97,110,105,115,99,104,32,35
90 DATA 35,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,
        32,32,32,32,32,32,32,35
100 DATA 35,32,42,83,99,104,114,105,102,116,97,114,116,32,32,62,110,60,111,114,
         109,97,108,32,32,32,32,32,32,32,35
110 DATA 35,32,32,32,32,32,32,32,32,32,32,32,32,32,32,62,107,60,111,109,112,
         114,105,109,105,101,114,116,32,32,35
120 DATA 35,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,
         32,32,32,32,32,32,32,35
130 :
140 DATA 35,32,42,68,114,117,99,107,101,114,32,32,32,32,32,62,114,60,117,101,
         99,107,115,101,116,122,101,110,32,32,35
150 DATA 35,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,
         32,32,32,32,32,32,32,35
160 DATA 35,32,42,32,32,32,32,32,32,32,32,32,32,32,32,62,98,60,101,101,110,100,
         101,110,32,32,32,32,32,32,35
170 DATA 35,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,
         32,32,32,32,32,32,32,35
180 DATA 35,32,45,45,45,62,103,101,119,97,101,104,108,116,58,62,32,60,32,32,32,
         32,32,32,32,32,32,32,32,32,35
190 :
200 DATA 35,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,
         32,32,32,32,32,32,32,35
210 DATA 35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35,
         35,35,35,35,35,35,35,35
220 :
230 DEFSTR Z,C:DEFINT I,S,K
240 DEF FNPK$(L,SP)=CHR$(27)+CHR$(127+L)+CHR$(127+SP)
250 CLS=CHR$(12)
260 DIM WF(17,31)
270 SRS=0:SRA=0
280 :
290 PRINT CLS;:RESTORE
295 PRINT FNPK$(1,1);"Demonstrationsprogramm"
300 PRINT FNPK$(4,1);"Bitte etwas warten!"
301 FOR K1=0 TO 16
302    FOR K2=0 TO 30
303       READ WF(K1,K2)
304    NEXT K2
305 NEXT K1
310 PRINT FNPK$(4,1);"Grundmenue:            "
320 PRINT FNPK$(6,1);"* >D<ruckersteuerung"
330 PRINT FNPK$(8,1);"* >B<eginn der Abarbeitung"
340 PRINT FNPK$(10,1);"--->gewaehlt:";
350 TA$="";TASP$=""
351 WHILE TA$<>"D" AND TA$<>"B"
353    TA$=INKEY$
355 WEND
360 WHILE TA$<>"B"
370    WHILE TA$="D"
380       PRINT FNPK$(10,25);"Druckersteuerung";
385       SRB=0:PRINT CHR$(131);
390       GOSUB 2000
400       WHILE SRB=0
460          TASP$=TA$
470          TA$=INKEY$:IF TA$="" THEN GOTO 470
480          WHILE (TA$="d" OR TA$="a" OR TA$="n" OR TA$="k" OR TA$="r" OR TA$="
b"):' AND SRB=0
500             PRINT FNPK$(10,28);TA$
510             IF TA$="d" THEN SRS=1
```

```
520             IF TA$="a" THEN SRS=2
530             IF TA$="n" THEN SRA=1
540             IF TA$="k" THEN SRA=2
550             IF TA$="r" THEN SRA=3:SRS=0
560             IF TA$="b" THEN SRB=1:TA$=""
565             SRBW=SRB
570             WHILE SRBW=0
580                TA$=INKEY$:IF TA$="" THEN GOTO 580
590                SRBW=1
600             WEND
610          WEND
612       WEND
615       PRINT FNPK$(18,28);" ";
620       GOSUB 2000
625       PRINT CHR$(130);
626       PRINT FNPK$(10,25);"                              "
627       PRINT FNPK$(10,25);
630       TA$=TASP$
640       TA$=INKEY$:IF TA$="" THEN GOTO 640
650    WEND
655    TA$=INKEY$:IF TA$<>"B" THEN GOTO 655
660 WEND
665 PRINT FNPK$(10,25):"Beginn der Abarbeitung"
670 SRBW=SRB
680 WHILE SRBW=1
690    PRINT FNPK$(13,1):"---> Bitte Drucker bereitmachen"
700    PRINT FNPK$(14,1):"---> Taste <W< betaetigen":
710    IF INKEY$<>"W" THEN GOTO 710
720    LPRINT:PRINT FNPK$(15,27):"* Drucker ist bereit"
730    SRSW=SRS:SRAW=SRA
740    LPRINT CHR$(27)+"@";
750    WHILE SRAW=3
760       PRINT FNPK$(17,27);"* Drucker ist initialisiert"
770       SRAW=0
780    WEND
790    WHILE SRSW=1
800       LPRINT CHR$(27)+"R"+CHR$(2);
810       PRINT FNPK$(17,27);"* deutscher Schriftsatz        "
820       SRSW=0
830    WEND
840    WHILE SRSW=2
850       LPRINT CHR$(27)+"R"+CHR$(0);
860       PRINT FNPK$(17,27);"* amerikanischer Schriftsatz"
870       SRSW=0
880    WEND
890    WHILE SRAW=1
900       PRINT FNPK$(18,27);"* normale Schriftbreite"
910       SRAW=0
920    WEND
930    WHILE SRAW=2
940       LPRINT CHR$(27)+CHR$(15);
950       PRINT FNPK$(18,27);"* Schmalschrift          "
960       SRAW=0
970    WEND
971    SRBW=0
972    :
973    SRSW=SRS
974    WHILE SRSW=1
975       LPRINT "Testausdruck f}r deutschen Zeichensatz"
976       LPRINT "---------------------------------------------":SRSW=0
977    WEND
978    WHILE SRSW=2
979       LPRINT "Testausdruck fuer amerikanischen Zeichensatz"
980       LPRINT "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
981       SRSW=0
982    WEND
990 WEND
1990 END
2000 FOR K1=4 TO 20
2010    FOR K2=12 TO 42
2020       BWS=PEEK(63488!+80*(K1-1)+(K2-1))
2030       SWAP BWS,WF((K1-4),(K2-12))
2040       POKE (63488!+80*(K1-1)+(K2-1)),BWS
2050    NEXT K2
2060 NEXT K1
2070 RETURN
```


Literatur

/1/ Mühlhaus, D.: Ausdrucken von Bildschirminhalten beim PC 1715 unter SCP. Mikroprozessortechnik 2 (1988) 3, S. 95

/2/ Gralla, D.: Zum Artikel „Ausdrucken von Bildschirminhalten beim PC 1715 unter SCP". Mikroprozessortechnik, Berlin 2 (1988) 7, S. 195

Plottersimulation mit KC 85/3 in Forth

In /1/ wurde für die Erläuterung des Programmierens in Forth das Beispiel einer einfachen Plotteransteuerung gewählt. Viele Leser werden, so wie wir, keinen Plotter besitzen und dennoch solche Beispiele ausprobieren wollen. Im folgenden Beitrag wird gezeigt, wie sich die vier Plotterfunktionen

- Stift anheben
- Stift senken
- Position anfahren
- zwei Punkte verbinden

auf dem Bildschirm simulieren lassen. Als Ausgangspunkt verwendeten wir den grafikfähigen Kleincomputer KC 85/2 bzw. KC 85/3 mit dem Forth-Modul M 026 oder mit geladenem Forth.

Zunächst wird in Screen 1 der Wortschatz des KC-Forth erweitert, wonach eine direkte Übernahme der Beispiele möglich wird. Danach werden die Variablen X_POS und Y_POS eingeführt, die die von einem echten Plotter automatisch beim Stehenbleiben gespeicherte aktuelle Position des Stiftes aufnehmen sollen. Aus analogen Gründen wird zur Unterscheidung, ob der Plotterstift gehoben oder gesenkt ist, die Variable STIFT eingeführt, wobei STIFT = 1 gesenkt und STIFT = 0 gehoben bedeutet.

Die gerätespezifischen Koordinatengrenzen sind in den Konstanten X_MIN, X_MAX, Y_MIN und Y_MAX abgelegt, wodurch die Übertragung auf andere Bildschirmgrößen einfach wird.

Zum Zeichnen von Strecken wird das im KC-Forth vorhandene Wort PLOT benutzt. Es erwartet fünf Parameter auf dem Datenstapel mit folgenden Bedeutungen

PLOT (f x y dx dy -- f x1 y1)

f Flag mit
f = 0 Linie wird gelöscht
f = 1 Linie wird gezeichnet

x, y Koordinaten des Ausgangspunktes
dx, dy Länge der x- bzw. y-Komponente
x1, y1 Koordinaten des Endpunktes

Das FLAG f wird in die gleichnamige Variable gesteckt, die mit 1 initialisiert wird, da ein Plotter nicht radieren kann. (In einer Ausbaustufe kann dieses Flag für die Funktion LOESCHEN verwendet werden.)

Die letzten beiden Parameter sind für das Beispiel in /1/ unpassend, weil dort statt der Koordinatendifferenzen die Absolutpositionen eingegeben werden müssen. Die Anpassung wird im Wort ZEICHNEN erledigt, das außerdem noch das Flag f unter die bereits auf dem Stapel befindlichen vier Werte mit Hilfe des Wortes FLAG_SETZEN legt und das Flag nach dem Zeichnen wieder vom Stack entfernt.

Nun haben wir alle Worte beieinander, die uns den Bildschirm als Plotter erscheinen lassen und können darangehen, die Plotterfunktionen aufzuschreiben. Beginnen wir mit den Worten HEBEN und SENKEN, die einfach die Variable STIFT entsprechend setzen.

Das grundlegende Wort ANFAHREN muß die aktuelle Position, in der sich der Plotter gerade befindet, als zusätzliche Information dem „gedächtnislosen" Bildschirm mitteilen. Sie wird als erstes auf den Stack geschafft und mit den Endpunktkoordinaten vertauscht, damit die Reihenfolge für ZEICHNEN stimmt. Bei der weiteren Abarbeitung des Befehls ANFAHREN wird mit der Variablen STIFT zwischen abgehobenem und abgesenktem Plotterstift unterschieden. Bei STIFT = 1 wird die Linie gezeichnet und bei Stift = 0 lediglich die sonst durch ZEICHNEN veränderte Stackbelegung korrigiert. Am Ende müssen die neuen Koordinaten den Positionsvariablen X_POS und Y_POS zugewiesen werden. Das Wort VERBINDEN kann danach sofort hingeschrieben werden und dokumentiert sich selbst.

Im 4. Screen sind einfache Beispiele für die Konstruktion weiterer Plotterbefehle aufgeführt. Unser Bildschirmplotter hat jetzt denselben Befehlsumfang und gleiche Schnittstellen wie der Plotter in /1/.

Das Vorgehen kann als Beispiel für die Anpassung eines konkreten Gerätes an eine vereinheitlichte Schnittstelle angesehen werden. Alle darauf aufbauenden Worte werden auf allen Geräten in gleicher Weise ablaufen. Das Programm stellt in dieser Form eine einfache Grundstufe für das Zeichnen von geradlinigen Figuren dar. Ausbaustufen zum Zeichnen von Ellipsen, Kreisen und komplizierteren Figuren sind möglich. Dazu vergleiche man auch /2/ und /3/.

Literatur

/1/ Pfüller, H. u. a.: **Einführung in Forth 83**, Teil 1. MP, Berlin 3 (1989) 3, S. 79

/2/ Jorke, G.; Lampe, B.; Wengel, N.: Arithmetische Algorithmen der Mikrorechentechnik. Berlin: Verlag Technik 1989

/3/ Taege, P.: Schnelle Algorithmen für Geraden und Kreise in Forth. MP, Berlin 3 (1989) 8, S. 246

Torsten Schmidt, Marco Thäle


```
Datei A:PLOTS.CF2       Block 1
0 ( KC-FORTH erweitern )
1
2 : 2SWAP  ( x1 y1 x2 y2 -- x2 y2 x1 y1 )
3      ROT  >R  ROT  R> ;
4 : 2OVER  ( x1 y1 x2 y2 -- x1 y1 x2 y2 x1 y1 )
5      >R >R  2DUP  R> R>  2SWAP ;
6 : 2DROP  ( x y -- )      DROP  DROP ;
```

```
Datei A:PLOTS.CF2       Block 2
0 ( PLOTTER  vorbereiten )
1
2  0 VARIABLE X_POS     0 VARIABLE Y_POS
3  0 VARIABLE STIFT
4  0 CONSTANT X_MIN     320 CONSTANT X_MAX
5  0 CONSTANT Y_MIN     256 CONSTANT Y_MAX
6
7  1 VARIABLE FLAG
8 : FLAG_SETZEN  ( x1 y1 x2 y2 -- f x1 y1 x2 y2)
9      >R >R >R >R  FLAG @  R> R> R> R>  ;
10
11 : ZEICHNEN  ( x1 y1 x2 y2 -- x2 y2 )
12     2OVER  >R  ROT  SWAP  -
13     SWAP  R>  -  FLAG_SETZEN  PLOT
14     ROT  DROP  ;
```

```
Datei A:PLOTS.CF2       Block 3
0 ( GRUNDBEFEHLE )
1
2 : HEBEN     0  STIFT !  ;
3 : SENKEN    1  STIFT !  ;
4
5 : ANFAHREN ( x y -- )
6     X_POS @  Y_POS @  2SWAP
7     STIFT @
8       IF  ZEICHNEN
9       ELSE  2SWAP  2DROP
10      ENDIF
11    Y_POS !  X_POS !  ;
12
13 : VERBINDEN ( x1 y1 x2 y2 -- )
14     HEBEN  ANFAHREN  SENKEN  ANFAHREN  ;
```

```
Datei A:PLOTS.CF2       Block 4
0 ( PLOTTER-ANWENDUNG )
1
2 : NULLPUNKT    X_MIN  Y_MIN  ;
3
4 : ACHSEN      ( -- )
5     HEBEN  X_MAX  Y_MIN ANFAHREN
6     SENKEN  NULLPUNKT  ANFAHREN
7     X_MIN Y_MAX  ANFAHREN  ;
```

Börse

Numerikpaket für comFORTH 2

Die durch den Standard Forth-83 vorgeschriebene Integerarithmetik erschwert oft die Lösung komplexerer Probleme aus dem wissenschaftlich-technischen Bereich. Für das System comFORTH 2 wurde für diesen Anwenderkreis ein geschwindigkeitsoptimiertes Numerikpaket mit folgenden Komponenten entwickelt:
– *Fließpunktarithmetik* für die Prozessoren Z 80, 8086 und 8087 in verschiedenen Genauigkeiten (4- und 10-Byte-Format) mit einem separaten Stapel gemäß dem Arbeitsdokument des ANSI-Standard-Teams; der konfigurierbare Wortschatz umfaßt Stapel- und Speicherbefehle, Grundrechenarten, Vergleichsbefehle, transzendente und Winkelfunktionen.
– *Konvertierungsmodul* zur Wandlung von Fließpunktzahlen in Zeichenketten (Ausgabeformate FIX, FLT, SCI und ENG)
– *Erweiterung des Systemdebuggers*
– *Vektor- und Matrixmodul* mit mächtigen Befehlen zur arithmetischen Verknüpfung von Vektoren und Matrizen
– Modul für *Komplexe Arithmetik* auf der Grundlage der Fließpunktarithmetik.
Bitte fordern Sie weiteres Informationsmaterial an.

Wilhelm-Pieck-Universität Rostock, Sektion Technische Elektronik, WB Automatische Steuerungen, Albert-Einstein-Straße 2, Rostock, 2500; Tel. 40 52 02 *Dr. Pfüller*

DUESE-Erweiterungen

Das in Heft 3/89, Seite 91, vorgestellte Softwarepaket **DUESE** wurde um zwei Programme für den EC 1834 erweitert, die jeweils einen Primäradapter ASK (ASC) mit den Schnittstellen V.24/IFSS unterstützen. Das Programm für die V.24-Schnittstelle wurde erfolgreich auf einem PC 1640 erprobt. In Vorbereitung sind außerdem noch Programme für den PC 1715 und für eine V.24-Schnittstelle des K 8915 (ASV).
Alle Bestandteile des Programmpaketes DUESE ermöglichen neben dem Dialogbetrieb die Angabe von Parametern in der Kommandozeile, was automatischen Betrieb möglich macht.

Ostthüringer Molkereikombinat, Industriestraße 1, Gera, 6500; Tel. 5 12 88 (Koll. Schumann) *Felgner*

Kopplung PC – K 1520

Die zu Steuerungszwecken entwickelte Software gestattet die Kopplung eines Rechners auf K 1520-Basis (Slave) und eines beliebigen Bedienrechners (Master) über IFSS- oder V.24-Interface, wobei die Parallelarbeit beider Rechner unterstützt wird. So können Kommandos mit ggf. benötigten Parametern aktiviert werden. Jeder Datentransfer wird, um eine hohe Funktionssicherheit zu garantieren, auf seine Richtigkeit überprüft. Die slaveseitige Kopplungssoftware besteht aus einem residenten Minimalmonitor (0,5 KByte) mit universellem Charakter. Er ermöglicht folgende Funktionen, die sich zur Lösung von Steuerungsaufgaben als ausreichend erwiesen haben: Speicher lesen und schreiben, Port lesen und schreiben, Start Anwenderprogramm.
Ein „Anwenderprogramm" kann jedes im K 1520 befindliche Unterprogramm sein. Die modular aufgebaute Master-Software enthält Funktionen, mit denen im Slave operiert werden kann: Slave synchronisieren, Einzelzeichen lesen und schreiben, Speicherbereich lesen und schreiben, Port lesen und schreiben, Unterprogramm starten, Unterprogramm-Endetest, Speicherbereich auf Diskette schreiben und von Diskette lesen.
Die Master-Software liegt in C sowie in Turbo-Pascal vor und wird unter CP/M-80, CP/M-86 und MS-DOS genutzt.

Akademie der Wissenschaften, Zentralinstitut für Optik und Spektroskopie, Abt. 9.4, Rudower Chaussee 6, Berlin, 1199; Tel. 6 74 37 55 *Feierabend*

Ein- und Ausgabe von KRS 4200-Dateien und Menügenerator

Die UNIX-Kommandos KRSIN und KRSOUT lesen bzw. schreiben Dateien im KRS 4200-Format. Dabei erfolgt eine Konvertierung zwischen dem KRS-Maschinencode (sogenannter 4/6/6-Code) und dem Byte-Code. Wahlweise kann das beim KRS-Textformat übliche NEWLINE (1EH) in LINE FEED (0AH) umgewandelt werden. Die Programme dienen vor allem zum lesenden und schreibenden Verarbeiten von KRS-Magnetbanddateien auf z. B. SKR- (oder auch P8000-)Anlagen unter UNIX. Beide Kommandos können sowohl „organisiertes Format" vom KRS, d. h. mit vorangestelltem Block- und Dateinummernzähler (üblich z. B. bei den Betriebssystemen SOET 4200, UEAS 4200) als auch freie Formate (physischer Blockinhalt gleich logischer Blockinhalt) in allen frei wählbaren Block- und Satzlängen verarbeiten. Die physische Umsetzung der Plattendateien auf ein KRS-Magnetband und umgekehrt erfolgt zweckmäßigerweise mit dem UNIX Standardkommando dd (1). Optimal lassen sich diese Programme in Kommandoprozeduren, den sogenannten Shell-Scripts zusammenfassen und ergeben so für jeden konkreten Anwendungsfall die maßgeschneiderte Lösung mit geringstem Aufwand. Die Entwicklung erfolgte in der Sprache C auf einem SM 1420. Die Portierungen zu WEGA und XENIX-286 brachten keine Probleme mit sich.
Die Benutzeroberfläche von UNIX zeichnet sich durch eine extrem hohe Flexibilität aus. Diese Flexibilität wird vom Softwareprofi sehr geschätzt, ist aber keine gute Oberfläche für den sogenannten Endnutzer. Für die einfachen Anwender erfreuen sich visuell orientierte, mit einfachen Benutzerdialogen ausgerüstete Benutzeroberflächen großer Beliebtheit. Um einerseits den Programmieraufwand für solche Menüs zu minimieren und andererseits Anreiz zu bieten, modular zu projektieren, wurde **MGen** entwickelt. MGen erzeugt aus seiner Eingabe-(Text-)datei ein komplettes C-Programm und läßt dieses compilieren und binden. Das Eingabefile enthält mindestens den Aufbau des Menübildes für den Anwender sowie eine Liste von Aktionen, in der spezifiziert wird, welche Aktivität (Kommando, Kommandofile oder Gruppe von Kommandos) nach einer konkreten Nutzereingabe erfolgen soll. Für Verwaltungszwecke darf die Eingabedatei noch einen beliebig langen Kommentarteil enthalten.
Der Menügenerator wurde auf einer SM 1420 unter dem Betriebssystem 2.9BSD-UNIX entwickelt und bereits auf das P8000-System unter WEGA übertragen. Er dürfte auf allen UNIX-Systemen lauffähig sein, sofern das System über den termcap(5)- oder terminfo(5)-Mechanismus verfügt. Eine Nutzung auf Systemen ohne diesen Mechanismus (z. B. MUTOS-Systeme) wird etwas problematisch sein, ist aber prinzipiell möglich, indem eine entsprechende Datei /etc/termcap angelegt, die Umgebungsvariable TERM gesetzt und die entsprechende Bibliothek übernommen wird. Der Generator wird zur Nachnutzung vollständig im Quellcode einschließlich kompletter Dokumentation (Beschreibung, Installation und Manual) ausgeliefert.

Bergakademie Freiberg, ORZ, Abteilung Systemprogrammierung/Basissoftware, Bernhard-von-Cotta-Straße 1, Freiberg, 9200; Tel. 51 28 43 oder 51 20 42 *Weber*

FastSearch Version 3.x

FastSearch ermöglicht die Erfassung der üblichen bibliographischen Daten von Zeitschriftenartikeln, Büchern, Patenten usw. und die Recherche nach den erfaßten Angaben. Die neue Version weist die Vorzüge der inzwischen verbreiteten Version 2.x auf (siehe MP 3/1989, Seite 90). Einige Details des Programms wurden wesentlich verändert oder erweitert.
1. Die Recherche ist jetzt nach unbegrenzt vielen Suchbegriffen (Autor, Schlagwort, Jahr, Titelwort ...) in beliebiger logischer Verknüpfung möglich.
2. Jegliche Änderungen der Datenbank werden sofort auf dem externen Datenträger (vorzugsweise eine Festplatte) sichergestellt. Damit wird eine sehr hohe Datensicherheit gegenüber ungewollten Störeinflüssen (z. B. Netzausfall) erzielt.
3. Groß- und Kleinschreibung sowie die Nutzung landesspezifischer Zeichen (vor allem der deutschen Umlaute) sind uneingeschränkt möglich.
4. Die Datenübernahme aus Volltextdatenbanken ist möglich. Zur Nutzung des Programms ist ein MS-DOS kompatibler Rechner mit möglichst 512 KByte Hauptspeicher erforderlich. Eine Festplatte ist nicht unbedingt erforderlich, wegen der sehr häufigen Zugriffe auf den externen Datenspeicher jedoch empfehlenswert. Interessenten können kostenlos eine Demonstrationsversion einschließlich der vollständigen Programmbeschreibung erhalten. Bei Bedarf kann auf einer zweiten Diskette auch eine Demonstrationsdatenbank mit etwa 1300 Einträgen zur NMR-Spektroskopie geliefert werden. Weiterhin ist während der Leipziger Frühjahrsmesse 1990 eine persönliche Konsultation im Hochhaus der Karl-Marx-Universität möglich.

Karl-Marx-Universität Leipzig, Sektion Chemie, Talstraße 35, Leipzig, 7010; Tel. 7 16 53 95/4 81 *Dr. Haeßner*

Adressenformatierung

Für eine textprogrammunabhängige und universell nutzbare Formatierung von Absender- und Adressatenangaben wurde das Programmpaket **ADF** entwickelt. Es beinhaltet Schnittstellen zur Anfertigung von Adreßlisten für Serienbriefe und zur Verwaltung der Absender- und Adressatenangaben über REDABAS oder gleichartige Verarbeitungssoftware. Gegenwärtig werden folgende Formatierungen ermöglicht:
– Mitteilungen ohne Vordruck
– Schreiben auf Kopfbogen.
Zeilenvorschübe können durch den Nutzer über eine Datei eingestellt werden. Die Pflege der Absender- und Adressatenangaben kann mit jedem beliebigen Editor oder Textprogramm erfolgen. Die Programme des Programmpaketes ADF können für jeden beliebigen Rechner mit einer Verarbeitungsbreite von 8 Bit aufwärts bereitgestellt werden. Gegenwärtig sind folgende Compilierungen verfügbar:
– P8000-Rechner mit WEGA-Betriebssystem und Textprogramm WEGA WORD
– 8-Bit-Rechner mit CP/M-kompatiblen Betriebssystemen und Wordstar-kompatiblen Textverarbeitungssystemen
– 16-Bit-Rechner mit MS-DOS-kompatiblen Betriebssystemen und Wordstar-kompatiblen Textverarbeitungssystemen.
Die mit ADF-Programmen vorgefertigten Mitteilungen oder Schreiben können mit dem jeweiligen Textverarbeitungssystem fertiggestellt werden.

Ministerium für Land-, Forst- und Nahrungsgüterwirtschaft, Bereich Pflanzen- und Tierproduktion, Bereichsverantwortlicher Rechentechnik, Köpenicker Allee 39-57, Berlin, 1157; Tel. 5 05 34 51 *Roscher*

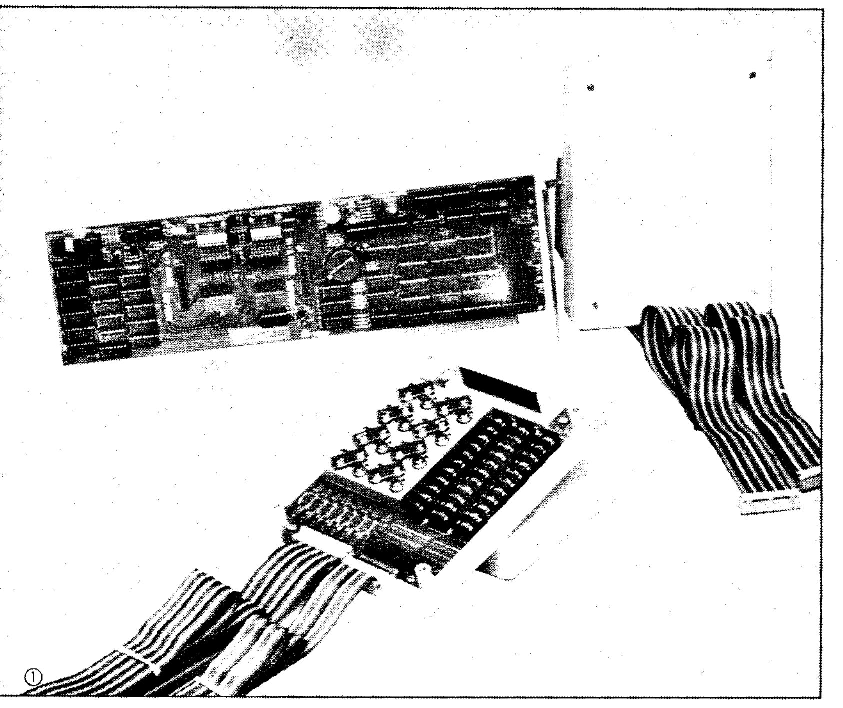
Wir suchen ...

... leistungsfähige Kombination von Kostenarten-, Kostenstellen- und Kostenträgerrechnung, die auf KBR A 6402 sowie kompatiblen Geräten wie SM 4 oder 5, K 1600-Baureihe oder auf rumänischen Geräten wie I 100 lauffähig ist. Angebote bitte an:

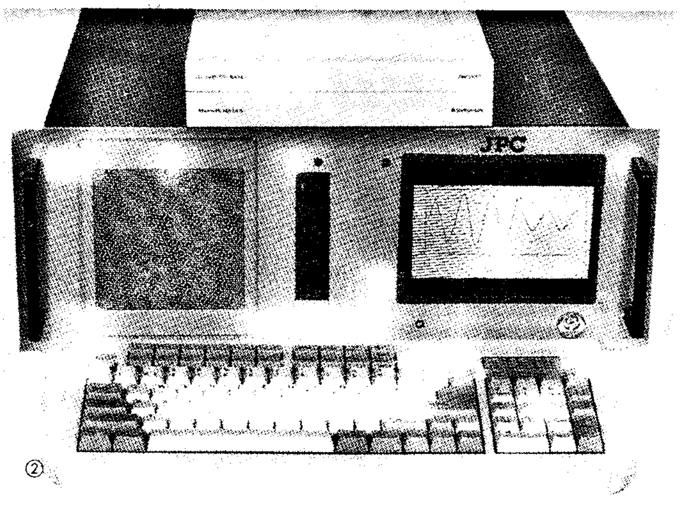
VEB Plastverarbeitungswerk Schwerin-Sacktannen, Hauptbuchhaltung, Schwerin, 2767; Tel. 45 25 35 oder 45 22 43 *Regner*

INTERKAMA 89

Die 11. INTERKAMA präsentierte sich wiederum als der weltweite Innovationsmarkt Nr. 1 für das Messen und Automatisieren. Details bieten auch andere Messen. Alle Details zu einem Ganzen summiert zu haben ist das Besondere an der INTERKAMA.
Vom 9. bis 14. Oktober 1989 wurde auf dem Messegelände in Düsseldorf (BRD) in 12 Messehallen mit 105 Ausstellerseminaren und durch 64 Fachvorträge (Kongreß 11.–12. 10.) der Weltstand auf diesem Fachgebiet eindrucksvoll demonstriert. Aus der vielschichtigen Palette sollen hier nur das Gebiet der PC-gestützten Systeme zur Labor- und Prüffeldautomatisierung sowie spezielle CAD-Systeme etwas näher beleuchtet werden, ohne dabei einen Anspruch auf Vollständigkeit zu erheben.
Generell ist einzuschätzen, daß der Weltstand auf dem Gebiet der PC-Nutzung gewaltig fortgeschritten ist. Sicher ist das erreichte Niveau der Eigenentwicklungen in der DDR beachtenswert und für den Industrie- und Routineeinsatz erforderlich bzw. ausreichend. Auf dem Gebiet der Grundlagen- und angewandten Forschung (AdW- und Hochschulbereich) bedeutet die Anwendung eigener, nicht dem Weltstand entsprechender Rechentechnik einen erheblichen Zeitverlust und die weitere Vergrößerung des Abstandes zum Weltniveau für zu untersuchende Methoden und Verfahren.


①

PC-gestützte Systeme zur Labor- und Prüffeldautomatisierung
Das Angebot von Einschubkarten für bzw. die Orientierung auf IBM-XT/AT und kompatible PCs sowie PS/2-Rechner ist weiter angewachsen. Mit dem von verschiedenen Herstellern angebotenen Spektrum kann das gesamte Gebiet der Labor- und Prüffeldautomatisierung befriedigt werden. So bietet die Firma *Ziegler-Instruments* ein umfangreiches Paket für Meßwerterfassungs-Systeme an. Hervorzuheben sind Module für die Meßwerterfassung bis 10 000 Messungen/s, Hochgeschwindigkeits-Meßwerterfassung (Transientenrecorder bis 200 kHz, Anti-Aliasing-Filter) und Signalgenerierung (Bild 1). Bei jedem Modul ist die hardwarebezogene Software bereits im Lieferumfang und Preis enthalten.
Ähnlich ist das Angebot der Firma *DATALOG* zu werten. Hier wird neben den erwähnten OEM-PC-Karten und einigen Low-Cost-Erfassungssystemen auch ein Datenerfassungs- und Prozeß-Interface (COBRA 700) als Beistellgerät angeboten. Aus der Kombination des hochintelligenten und flexiblen Beistellsystems mit dem leistungsfähigen PC resultiert ein, hinsichtlich Leistungsfähigkeit, Anpassungsflexibilität und Anwendungsfreundlichkeit, überlegenes System. Die Aufgaben Meßwerterfassung, Prozeßkontrolle, Datenverarbeitung und Test-Applikation in Industrie, Labor und Forschung sind damit lösbar.
Von der Firma *disys* wird besonderer Wert auf die Ausrüstung von Industrie-Prozeßrechnern (IPC, basierend auf XT/AT) gelegt. Spezieller Vorzug ist die Anpassung an die industrielle Umwelt (Robustheit, Netzspannungsfilterung, Optoentkopplung, Überdruckbelüftung) (Bild 2). Dabei bleibt jedoch die volle Kompatibilität zur großen Welt der MS-DOS-Software gewahrt.
Siemens dagegen bietet bezüglich der PC-Meßtechnik vorrangig komplette Applikationslösungen (z. B. Reglergeräte-Prüfung, Netzteilprüfung, Tiefpaß-Kenngrößen-Aufnahme u. v. a.) an. Darüber hinaus werden von einer Vielzahl von Herstellern Zusatz- und Peripheriegeräte für PC-gestützte Grundsysteme angeboten. Stellvertretend für viele soll hier das Angebot von speziellen Schreibern, Plottern und Druckern der Firma *LINSEIS* oder IBM-kompatible Datenspeicher mit Tracker-Serie (1,2 Mbyte) von *Ziegler* erwähnt werden.
Ebenso vielfältig wie die oben dargestellten Hardwaresysteme zeigen sich die angebotenen Softwaremodule, meist als integraler Systembestandteil. Neben den MS-DOS-Betriebssystemmodulen und Benutzeroberflächen bestimmen in erster Linie Softwarepakete zur Meßdatenanalyse (Erfassung, Ausgabe, Speicherung, Auswertung und Archivierung) das Angebot. Das von *Ziegler* entwikkelte Paket signalys zeichnet sich z. B. durch seine weitgehende Hardware-Unabhängigkeit aus, es läuft auf insgesamt 20 verschiedenen PCs mit unterschiedlichsten Graphik-Controllern. Ein breites Feld nimmt die graphische Meßdaten-Auswertung, einschließlich der Window-Technik, ein (z. B. DATALOG).
Eine außerordentlich leistungsfähige technisch-wissenschaftliche PC-Software ist das Programmsystem ASYST 3.0 von *KEITHLEY*. ASYST paßt sich der Hardware, den Meßwerterfassungsarten und den Erfordernissen für Analyse-Operationen an. Einzelne ASYST-Kommandos leisten das, wofür man in C, Fortran oder Basic ganze Routinen benötigt (menügeführt). ASYST kann praktisch mit jeder Geschwindigkeit Daten erfassen, die die Meßwerterfassungs-Hardware zuläßt. Ein umfangreicher Support zur Einarbeitung in das System steht zur Verfügung.



②

CAD-Systeme
Von besonderem Interesse waren die neuen Versionen des Systems CADdy von *Ziegler-Instruments*. Durch eine intensive, kontinuierliche Forschung und Weiterentwicklung stellt die Firma sicher, daß CADdy auf dem neuesten technischen Stand ist. Für viele Bereiche ist von besonderem Vorteil, daß das System neben den Branchen Elektronik und Konstruktion auch für Anlagenbau, Elektrotechnik, Vermessung und Architektur anwendbar ist. Die Elektronik-Branche realisiert alle für die Erstellung von Vorlagen zur Leiterplattenfertigung anfallenden Arbeiten (Schaltplan und Layout mit Stück- und Verbindungsliste sowie Autorouter und Post-Processing). Die Konstruktions-Branche zeichnet sich durch die 2D-Konstruktion, die 3D-Volumendarstellung und NC-Fertigungsunterlagenerstellung aus.
Eine wirksame Unterstützung der innovativen Elektronik-Technologien, Hybridschaltungs-Design und SMD-Layouts bietet das System BiCAD von *BINDER*. Funktionen wie automatisches Erfassen von Dickschichtpastenparameter, Widerstandsentwurf, thermische Simulation oder Postprozessor für Laser-Trim-Systeme sind enthalten.
Es soll nochmals betont werden, daß echte CAD-Systeme als Mindestkonfiguration über eine hochauflösende Farbgraphik verfügen. Versuche mit Hardwarekonfigurationen, die unter diesem Niveau liegen, werden kaum die erwarteten Rationalisierungseffekte erreichen.

Dr. F. Müller

Entwicklungen und Tendenzen

Anwendermodifizierbares PC-BIOS

Ein beim Anwender modifizierbares BIOS (Basic Input/Output System) für Personalcomputer auf AT-, MCA- oder EISA-Basis wollen Intel und die Phoenix Technologies, Ltd., gemeinsam entwickeln. Mit Hilfe von Ein-Megabit-Flashspeichern in ETOX-Technologie (EPROM Tunnel Oxide) sowie BIOS-Software und Dienstprogrammen von Phoenix können PC-Hersteller den BIOS-Code per Diskette modifizieren, ohne dazu das PC-Gehäuse öffnen und die BIOS-Chips austauschen zu müssen. Flashspeicher sind hochintegrierte, elektrisch löschbare, nichtflüchtige Halbleiterspeicher.
Laut Jonathan Joseph, General Manager der PC-Division von Phoenix, bietet ein modifizierbares BIOS den System-OEMs eine Reihe von Vorteilen: „PC-Hersteller werden Kosten einsparen, indem sie ihren Kunden neue BIOS-Versionen auf Disketten liefern, anstatt dazu einen Kundendiensttechniker abzustellen. Sie können auch die Zeit bis zur Markteinführung neuer Produkte dadurch verkürzen, daß sie den endgültigen BIOS-Code als letzten Fertigungsschritt – kurz bevor das System ausgeliefert wird – implementieren."
Und Dick Pashley, General Manager der Flash Memory Operation von Intel: „Die Kombination der Flashspeicher-Technologie mit der Möglichkeit, das BIOS zu modifizieren, verleiht den Anwendern moderner hochentwickelter PCs eine neue Dimension an Flexibilität. Anwender solcher Systeme werden in der Lage sein, ihre vorhandenen PCs an neue Peripheriegeräte, die BIOS-Unterstützung benötigen – beispielsweise Plattenlaufwerke, Monitore und Tastaturen – anzupassen, sobald diese verfügbar sind." *MP*

Amiga 2500/30 mit 25 MHz

Von Commodore wurde vor kurzem der neue Amiga, das Modell 2500/30, angekündigt, der mit dem 32-Bit-Prozessor Motorola 68 030 arbeitet. Die Taktfrequenz dieses Prozessors beträgt 25 MHz. Bereits in der Grundausstattung verfügt der Amiga über 2 MByte Hauptspeicher, den 25 MHz schnellen Koprozessor 68 882, den Speicherverwaltungs-Chip 68 851 sowie eine 40-MByte-Festplatte mit einem selbstbootenden SCSI-Controller. Auch der Amiga 2500/30 kann mit einer Brückenkarte ausgerüstet werden, die das Abarbeiten von MS-DOS-Programmen erlaubt. Der Rechner hat einen Video-, fünf Amiga-, zwei XT- und zwei AT-Steckplätze. Damit soll sich der neue Amiga laut Commodore besonders für professionelle Video- und Musikanwendungen eignen. Für die bestehende Amiga-2000-Familie wird zur Nachrüstung eine 2630-Acceleratorkarte angeboten. *MP*

Siemens und Nixdorf gemeinsam

Die Siemens AG, Berlin und München, und die Nixdorf Computer AG, Paderborn, werden auf dem Gebiet der Daten- und Informationstechnik künftig zusammenarbeiten.
Es wurde vereinbart, daß die Siemens AG – vorbehaltlich der Zustimmung der hierfür zuständigen Gremien und Behörden – die Mehrheit der Stammaktien der Nixdorf Computer AG erwirbt; die Siemens AG wird ihren Bereich Daten- und Informationstechnik in die Nixdorf Computer AG einbringen. Wegen der relativ geringen Überlappungen werden bei einem Zusammengehen gute Chancen gesehen, bedeutende Synergiepotentiale zu erschließen.
Das gemeinsame Unternehmen, das einen Umsatz von rund 12 Milliarden DM haben wird, soll den Namen SIEMENS-NIXDORF Informationssysteme AG führen.
Von der Nixdorf Computer AG wurde inzwischen die geplante Entlassung von 4880 Mitarbeitern bis zum 1. Juli dieses Jahres bekanntgegeben. *MP*

Fortschritte in der Halbleitertechnologie verdoppeln die Leistung der HP-PA-Computer

Mit VLSI-Schaltungen in Submikron-CMOS-Technologie will Hewlett-Packard die Leistung der RISC-basierenden Precision-Architecture-Computer (HP-PA) um 70 bis über 100 Prozent steigern. Die neuen ICs sind vorrangig für die Minicomputer- und Server-Familien bestimmt; ein Chipsatz für Workstations ist in Vorbereitung.
Mit 12 bis 16 Mio Gleitkomma-Operationen pro Sekunde (MFLOPS) und Prozessor soll die neue Halbleitertechnologie die Leistung der HP-Spitzenmodelle in Großrechner-Dimensionen bringen.
Kleine Abmessungen der Chips erleichtern das Kombinieren mehrerer CPUs zu symmetrischen Multiprozessor-Konfigurationen.
Die Submikron-CMOS-Implementierung, also Strukturen unter 1 μm, erlaubt Schaltungen extrem hoher Integrationsdichte zur Steigerung der Prozessorleistung. Dank dieser CMOS-Technologie liefert HP Großrechnerleistung mit einer Einchip-CPU. Vergleichbare Modelle mit CISC-Architektur benötigen dafür 100 und mehr Chips. *MP*

Echtzeit-Workstation für die Simulation

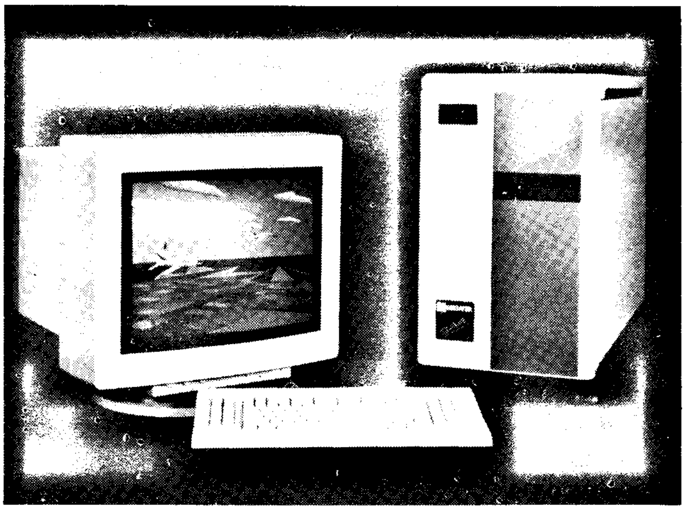
Mit der MEGATEK 944 (Bild unten) stellte CIS Graphik & Bildverarbeitung auf der SYSTEMS 89 eine neue Workstationgeneration für die Echtzeit-Simulation vor. MEGATEK ist ein Unternehmen der United Telecom und wird von CIS exklusiv vertreten. Durch die Integration von speziellen Prozessoren von Weitek, Texas Instruments und Motorola in einem Standard-VME-Bus können die Echtzeitanforderungen an 3D-Hochleistungsgrafik und -simulation erfüllt werden. *MP*

Super-VGA-Karte nach Standard

Ende 1988 taten sich auf Betreiben von NEC verschiedene Monitor- und Grafikkartenhersteller zusammen, um einen neuen Grafikstandard vorzuschlagen (s. MP 2/1989, S. 60). Ziel war die Standardisierung von Grafikauflösungen oberhalb des IBM-VGA-Bereiches, also über 640 × 480 Bildpunkten. Im Oktober veröffentlichte das VESA genannte Gremium (Video Electronics Standard Association) einen Standard für solche VGA- und 8514/A-kompatiblen Grafikkarten. Von Headland/Video Seven wurde nun die erste VESA-kompatible Karte vorgestellt, die VGA 1024i. Die Karte gibt es in einer 256-KByte- und einer 512-KByte-Version, und sie kann bei einer Auflösung von 1024 × 768 Bildpunkten vier Farben aus 262 144 Farbtönen darstellen. Zur Softwareunterstützung werden fünf Disketten mitgeliefert. *MP*

Perspektiven der Informationsindustrie

In Japan wurde durch das MITI eine Studie zu den Perspektiven der Informationsindustrie im Jahre 2000 herausgegeben. Danach wird die Informationsindustrie zur Leitindustrie der japanischen Wirtschaft. Bei einer jährlichen Wachstumsrate von 13 % – gegenüber einem Wachstum der Gesamtwirtschaft von 5 % – wird der Anteil der Informationsindustrie im Jahre 2000 20,7 % betragen. Informationsbezogene Investitionen der Unternehmen werden von 11 % auf 33 % steigen. Die Zahl der Arbeits-



Fragebogen *zur Ost-West-Kontaktbörse (siehe folgende Seite)*

Ich interessiere mich vorrangig für

- ☐ *Hobby-Anwendungen, Computerspiele*
- ☐ *allgemeine Anwendungen (Textverarbeitung, Datenbanken etc.)*
- ☐ *kaufmännisch orientierte Programme, Branchensoftware*
- ☐ *CAD- und CAE-Software*
- ☐ *Computer-Hardware, maschinennahe Programmierung*
- ☐ *technische Rechneranwendungen; Messen, Steuern, Regeln*
- ☐ *Programmentwicklung in*
  - ☐ *Assembler*
  - ☐ *Basic*
  - ☐ *Pascal*
  - ☐ *C*
  - ☐ *KI-Sprachen*
  - ☐ *sonstigen Sprachen*
- ☐ *theoretische Informatik*
- ☐ *Telekommunikation*

Zutreffendes bitte ankreuzen (Mehrfachnennungen sind möglich).
Meinen Kenntnisstand schätze ich etwa so ein:

- ☐ *1 (Ich bin Experte.)*
- ☐ *2*
- ☐ *3*
- ☐ *4*
- ☐ *5*
- ☐ *6 (Ich bin Anfänger.)*

(Bitte erteilen Sie sich eine „Schulnote".)
☐ *Ich bin auch bereit, mit mehreren Interessenten zu korrespondieren.*

plätze in der Informationsindustrie soll im Jahre 2000 4,1 Mio (1984: 1,5 Mio) betragen.
Zur Entwicklung der Informationsindustrie sind unterstützende Regierungsmaßnahmen auf den Gebieten des Datenschutzes und der Datensicherheit, Rahmenbedingungen für Telekommunikationsdienste sowie die Schaffung höherwertiger Informationsdienste, Maßnahmen zur Systemintegration und zur Austauschbarkeit zwischen Systemen erforderlich.

Quelle: Online. – Köln (1989) 11. – S. 69 *Wi*

Übersetzungssystem Japanisch – Deutsch

Die Firma Fujitsu entwickelte gemeinsam mit dem Stuttgarter Softwarehaus Aris ein Maschinenübersetzungssystem vom Japanischen ins Deutsche. Dieses System basiert auf dem seit 1985 auf dem Markt angebotenen Grundsystem Atlas II, das mittels einer Zwischensprache die Übersetzung von mehreren bzw. in mehrere Sprachen ermöglicht. Das System ist direkt an die japanische Textdatenbank von Nikkei Telecom angeschlossen und übersetzt Texte in Echtzeit ins Deutsche. Mit der Entwicklung des Systems wurde 1986 begonnen. Es läuft auf Workstations und bietet, z. B. auf einer Sun 4/110, eine Übersetzungsgeschwindigkeit von 6000 Wörtern pro Stunde. Dabei benutzt es ein Deutsch-Wörterbuch mit 50 000 Wörtern, und es stützt sich auf 12 000 Grammatikregeln.
Die Firma Fujitsu stellte für diese Entwicklung den Übersetzungsprozessor und die Komponente zur Grammatikanalyse zur Verfügung. Von der Firma Aris wurden die Forschungs- und Entwicklungsarbeiten für die deutsche Spracherzeugungskomponente bereitgestellt.
Bis zur Serienreife des Systems bedarf es noch anwendungsspezifischer Verbesserungen. Ein Probebetrieb mit ausgewählten Anwendern ist für Ende dieses Jahres geplant.

Quelle: Elektronik. – München 38 (1989) 18. – S. 7 *Fa*

Superrechner aus Südkorea

Das Forschungsinstitut Korea Advanced Institut of Science and Technology in Südkorea soll einen Superrechner entwickelt haben. Damit wäre Südkorea nach den USA, Japan und der BRD das vierte Land, in dem Superrechner entwickelt werden. Der südkoreanische Rechner soll eine Verarbeitungsgeschwindigkeit von 10 Mrd. Rechenoperationen pro Sekunde und eine Speicherkapazität von 256 MByte aufweisen.
Die zweijährige Forschungsarbeit wurde von der britischen Firma General Electric Co finanziell unterstützt. Man rechnet damit, daß der Superrechner im Februar 1990 vorgestellt wird.

Quelle: Eildienst. – Berlin (1989) 171. – S. 1 *Wi*

Lichtwellenleiter-Chip für 1 Gigabit pro Sekunde

Die Entwicklung von zwei experimentellen Chips zum Senden und zum Empfang von Daten über Lichtwellenleiter soll der Firma IBM gelungen sein; die Übertragungsgeschwindigkeit wird mit einer Milliarde Bit in der Sekunde angegeben.
Die Chips basieren auf Galliumarsenid. Der Empfänger – von der Größe eines Fingernagels – soll fünfzigmal mehr optische und elektronische Schaltungskomponenten enthalten, als bisher auf Chips für optoelektronische Empfänger bzw. Sender gepackt werden konnten. Der Chip soll mehr als 8000 Transistoren und Strukturdetails mit einer Strukturgröße von einem Mikrometer enthalten.
Diese Chips werden als Vorläufer von Sende- und Empfangschips für zukünftige ultraschnelle Lichtwellenleiterverbindungen großer Datennetze angesehen.

Quelle: Blick durch die Wirtschaft vom 15. 9. 1989 *Fa*

Strom in Leiterbahnen messen

Von der Firma Laplace Ltd. wurde für Leiterplatten ein Gerät zum Messen der Ströme in Leiterbahnen vorgestellt. Damit soll es möglich sein, die Messung auch ohne Auftrennen der Leitungen und bei aktivem Zustand der Schaltung durchzuführen. Auf die zu messenden Teilstücke der Leiterbahnen werden zwei Anschlußstifte gesetzt, über die das Gerät die jeweilige Stromflußrichtung zwischen zwei Punkten feststellt. Danach wird ein einstellbarer Strom in das Teilstück injiziert, der gegenläufig dem dort fließenden Strom ist. Erreichen beide ein Verhältnis von 10:1, so beginnt eine Signalverarbeitungsschaltung im Gerät, den Spannungsabfall im Teststück zu ermitteln. Dazu läßt sich der Stromwert (ohne den injizierten Strom) errechnen und anzeigen. Der Meßbereich geht von 1 mA bis 1 A, die Genauigkeit liegt bei 5 %.

Quelle: Elektronik. – München 38 (1989) 20. – S. 6 *Fa*

Transistor funktioniert wie ein Parallelprozessor

Die Wissenschaftler der AT & T Bell Laboratories haben einen neuen, kleinen Transistor vorgestellt, der 20mal leistungsfähiger sein soll als die herkömmlichen Bausteine dieser Art. Dieser *Multistate Resonant Tunneling Bipolar Transistor* könne nach Angaben von AT & T als kleiner Parallelprozessor eingesetzt werden. Damit sei man dem im vergangenen Jahr von Texas Instruments vorgestellten Baustein einen Schritt voraus. Beide Transistoren nutzen Tunneleffekt-Techniken und führen zur Entwicklung von Schaltkreisen, die wesentlich schneller und preiswerter sind als herkömmliche. Außerdem benötigen sie weniger Energie. In der Halbleiterentwicklung tätige Wissenschaftler gehen davon aus, daß derartige Bausteine, an denen auch die Firmen IBM und Fujitsu arbeiten, die Basis für eine neue Generation noch kleinerer und schnellerer Computerchips sein werden.

Quelle: VDI-Nachrichten vom 26. 7. 1989 *Fa*

Verbindung aufnehmen

Ein Aufruf an unsere Leser

Wer die Presse der letzten Zeit aufmerksam verfolgte, dem werden sicher nicht die zahlreichen Hinweise zur Unterstützung von deutsch-deutschen Kontakten, oft Beratungen auf kommerzieller Basis, entgangen sein. Wir halten auch solche Formen für nützlich und werden sie in unserer Zeitschrift fördern. Allerdings meinen wir, daß es in unserem Fachgebiet viele Interessierte gibt, die sich auch unentgeltlich über Klippen im Umgang mit der Computertechnik helfen wollen. Diese Form der grenzüberschreitenden Zusammenarbeit wollen wir mit einer Gemeinschaftsaktion der Redaktionen c't und MP unterstützen. Mit diesem Aufruf, der in ähnlicher Form in Heft 3/90 der c't erscheint, geben wir Ihnen die Möglichkeit, mit kontaktsuchenden Computeranwendern in der BRD Verbindung aufzunehmen. Wer also brieflich über Hard- und Software fachsimpeln, in einen direkten persönlichen Erfahrungsaustausch treten oder sich gegenseitig mit Hard- und Software helfen möchte, der fülle bitte den folgenden Fragebogen aus und sende ihn im Briefumschlag bis spätestens 15. April 1990 an die Redaktion c't.

Auf der Basis Ihrer Angaben und der eingegangenen Fragebögen aus der BRD werden per Computer Korrespondenzpartner mit übereinstimmenden Interessen zusammengeführt. Vergessen Sie deshalb bitte nicht, Ihr Einverständnis mit der EDV-gestützten Verarbeitung der Fragebögen durch Unterschrift zu bekunden. Die Daten werden ausschließlich für den beschriebenen Zweck verwendet.
Sollten mehrere Leser dieses Heftes Interesse an einer Partnerschaft haben, kann auch eine Kopie des Fragebogens eingesandt werden. Die c't-Redaktion hat zugesagt, jedem MP-Einsender per Brief den Namen und die Anschrift des vom Computer ermittelten BRD-Korrespondenzpartners mitzuteilen.

Viel Erfolg also beim Ost-West-Erfahrungsaustausch!

Ihre Redaktion MP

✂ ·

Bis 15. April 1990 einsenden an:

Redaktion c't
Kontaktbörse
Helstorfer Straße 7
D-3000 Hannover 61

Ich möchte persönlichen Kontakt zu Computeranwendern in der BRD aufnehmen. Ich bin damit einverstanden, daß meine Anschrift und die Angaben dieses Fragebogens zu diesem Zweck in einer Datenverarbeitungsanlage gespeichert und an Teilnehmer der Kontaktbörse weitergeleitet werden.

____
Datum *Unterschrift*

Absender (bitte deutlich schreiben)

____
Vorname/Name

____
Straße/Nr.

____
PLZ/Ort

____
Telefon

Notation

<> Teile, die in den Clipper-Kommandos vom Nutzer zu ersetzen sind.
[] Teile, die wahlweise angegeben werden können.
| Alternativen
{} wahlweise Wiederholung
SCHLÜSSELWÖRTER von Clipper-Kommandos werden in großen Buchstaben gedruckt.
Einige elementare Sprachkonstruktionen (Metasymbole) werden wegen ihrer Einfachheit nur verbal erklärt oder als selbsterklärend betrachtet:
<ausdr> Ausdruck
<bedingung> und <l_ausdr> logischer Ausdruck
<c_ausdr> Ausdruck vom Typ Zeichenkette (character, Zeichen)
<cm_>ausdr> Ausdruck vom Typ Zeichenkette oder Memo
<d_ausdr> Ausdruck vom Typ Datum
<n_ausdr> numerischer Ausdruck
<gb> Gültigkeitsbereich (NEXT <n_ausdr> oder RECORD <n_ausdr> oder ALL).

Kommandos

! | **RUN** <DOS_kommando> Aufruf eines DOS-Kommandos.
&& <text> Leitet einen Kommentar in Kommandozeilen ein.
***** | **NOTE** <text> Kommentarzeile mit Makroersetzung.
? [<ausdr_liste>] Übergang zur nächsten Zeile (cr, lf) und Ausgabe einer Folge von Ausdrücken auf dem Bildschirm.
?? <ausdr_liste> Ausgabe einer Folge von Ausdrücken auf dem Bildschirm, ohne vorherigen Zeilenwechsel (cr, lf).
@ <oben>,<links>,<unten>,<rechts> **BOX** <c_ausdr>
Zeichnen eines Rahmens auf dem Bildschirm. Die ersten neun Zeichen von <c_ausdr> definieren die Rand- und Füllzeichen, beginnend mit dem Zeichen für die linke obere Ecke.
@ <oben>,<links> **CLEAR** [**TO** <unten>,<rechts>] Löschen eines Rechtecks auf dem Bildschirm.
@ <zeile>,<spalte> **PROMPT** <c_ausdr_1> [**MESSAGE** <c_ausdr_2>]
Ausgabe eines Menütextes (c_ausdr_1) und eines Erläuterungstextes (c_ausdr_2) beginnend auf der angegebenen Position (vgl. SET MESSAGE und MENU TO).
@ <zeile>,<spalte>
[**SAY** <ausdr>[**PICTURE** <klausel>]]
[[**GET** <var>[**PICTURE** <klausel>]
[**RANGE** <ausdr_1>,<ausdr_2>]
[**VALID** <bedingung>]]
Ausgabe des berechneten Wertes von <ausdr> auf dem Bildschirm ab der angegebenen Zeile und Spalte. Anforderung eines Wertes für die Variable oder für das Feld nach GET. <klausel> enthält Formatfunktionen und -zeichen. RANGE definiert einen Gültigkeitsbereich für numerische und Datumswerte. VALID legt eine Bedingung fest, die die Werte erfüllen müssen (vgl. READ).
@ <oben>,<links> **TO** <unten>,<rechts> [**DOUBLE**]
Zeichnen eines Rahmens auf dem Bildschirm mit einfacher oder doppelter Umrandung (DOUBLE).
ACCEPT [< prompt >] **TO** <speichervar>
Eingabe einer Zeichenkette von der Tastatur.
APPEND BLANK
Anfügen eines Datensatzes an das Ende des aktiven Datenbankfiles.
APPEND FROM <filename> [<gb>] [**FOR** | **WHILE** <bedingung>]
[**FIELDS** <feldliste>] [**SDF**] | [**DELIMITED** [**WITH BLANK** | <zeichen>]]
Anfügen eines Datenbankfiles oder eines Textfiles (SDF) an das Ende des aktiven Datenbankfiles.
AVERAGE [<gb>] <ausdr_liste> **TO** <var_liste> [**FOR** | **WHILE** <bedingung>]
Berechnung des Mittelwertes numerischer Felder.

BOX Siehe @ ... BOX.
BEGIN SEQUENCE
[<kommandofolge>]
[**BREAK**]
[<kommandofolge>]
END Strukturierte Anweisung zur privaten Fehlerbehandlung. Die Abarbeitung von BREAK bewirkt die Übergabe der Programmsteuerung an das Kommando nach dem END.

CALL <extern_prog> [**WITH** <parameter_liste>]
Aufruf separat übersetzter Programme anderer Sprachen.
CASE Siehe DO CASE.
CANCEL | **QUIT** Abbruch der Programmausführung, Schließen aller Dateien und Rückkehr zum rufenden Programm.
CLEAR Löschen des Bildschirms und Deaktivieren der aktiven @ ... GET-Kommandos.
CLEAR ALL Abschließen aller in Benutzung befindlichen Files (Datenbank-, Index-, Formatfiles usw.), Freigabe aller Speichervariablen und Auswahl des Arbeitsbereichs 1.
CLEAR GETS Deaktivieren der aktiven @ ... GET-Kommandos.
CLEAR MEMORY Löschen aller Speichervariablen.
CLEAR SCREEN Löschen des Bildschirms.
CLEAR TYPEAHEAD Löschen aller Zeichen im internen Tastaturpuffer.
CLOSE [**ALL** | **ALTERNATIVE** | **DATABASES** | **FORMAT** | **INDEX**]
Abschließen der ausgewählten Files.
COMMIT Übergabe aller Clipper-Puffer an das Betriebssystem (DOS 3.2) und Ausgabe auf Platte (DOS 3.3).
CONTINUE Fortsetzen der Suche nach LOCATE.
COPY FILE <quell_filename> | (<c_ausdr>) **TO** <ziel_filename> | (<c_ausdr_2>)
Kopieren beliebiger Files.
COPY STRUCTURE TO <filename> | <c_ausdr>) [**FIELDS** <feldliste>]
Es wird ein neues Datenbankfile erstellt, mit der Struktur des aktiven Datenbankfiles und null Sätzen.
COPY STRUCTURE EXTENDED TO <strukturfile> | (<c_ausdr>)
Die Struktur des aktiven Datenbankfiles wird zum Inhalt des Datenbankfiles.
COPY TO <filename> | (<c_ausdr>) [<gb>] [**FIELDS** <feldliste>
[**FOR** | **WHILE** <bedingung>]
[**SDF** | **DELIMITED** [**WITH BLANK** | <zeichen>]]
Kopieren des aktiven Datenbankfiles in ein anderes Datenbank- oder Textfile.
COUNT [<gb>] [**FOR** | **WHILE** <bedingung>] **TO** <speichervar>
Anzahl der Datensätze.
CREATE <filename> |(<c_ausdr>) Erzeugen eines leeren Strukturfiles.
CREATE <filename> |(<c_ausdr_1>) Erzeugen eines leeren Strukturfiles.
CREATE <filename> |(<c_ausdr_1>) **FROM** <strukturfile>| (<c_ausdr_2>)
Erzeugen eines Datenbankfiles mit der im Strukturfile beschriebenen Struktur.

DECLARE <speichervar>[<n_ausdr>] {<speichervar> [<n_ausdr>] ...}
Deklaration eindimensionaler Felder. Beachte: Hier sind die eckigen Klammern keine Hilfssymbole! (DECLARE a[5]).
DELETE [<gb>] [**FOR** | **WHILE** <bedingung>]
Löschmarkierung für Datensätze setzen.
DELETE FILE <filename> |(<c_ausdr>) Löschen des angegebenen Files.
DIR [<gerät>:] [<pfad>\] [<filegruppe>| (<c_ausdr>)]
Anzeige der ausgewählten Filenamen.
DISPLAY [<gb>] [<ausdr_liste>] [FOR | **WHILE** <bedingung>] [**OFF**]
[**TO PRINT** | **TO FILE** <filename>| (<c_ausdr>)]
Ausgabe der berechneten Werte.
DO <filename> [**WITH** <parameter_liste>]
Aufruf eines Kommandofiles oder einer Prozedur.
DO CASE
CASE <bedingung>
<kommandofolge>
{**CASE** <bedingung>
<kommandofolge> ...}
[**OTHERWISE** <kommandofolge>]
ENDCASE
Strukturierte Anweisung zur Übergabe der Programmsteuerung an den ersten Fall, dessen Bedingung erfüllt ist.
DO WHILE <bedingung>
[<kommandofolge>]
ENDDO
Strukturierte Anweisung zur zyklischen Abarbeitung der Kommandofolge, solange die Bedingung erfüllt ist (siehe LOOP, EXIT).

EJECT Seitenvorschub auf dem Drucker.
ERASE <filename> | (<c_ausdr>) Löschen des angegebenen Files.
EXIT Verlassen eines DO-WHILE- oder eines FOR-Zyklus und Fortsetzung mit dem auf das zugehörige ENDDO folgenden Kommando.
EXTERNAL <proz_liste> Erzwungene Bezugnahme auf ein externes Symbol.
FIND <zeichenfolge> | (<c_ausdr>) Voraussetzung: Die aktive Datenbank ist mit einem Indexfile in Benutzung. Dann bewirkt FIND die Positionierung des Dateizeigers auf den ersten Datensatz, für den Indexschlüssel und berechnete Zeichenkette gleich sind.
FOR <speichervar> = <anfangswert> **TO** <endwert> [**STEP** <schrittweite>]
<kommandofolge>
NEXT Strukturierte Anweisung für die zyklische Abarbeitung der Kommandofolge in Abhängigkeit von Zählwerten (Laufanweisung).
FUNCTION <funk_name>
[**PARAMETERS** <par_liste>]
<kommandofolge>
RETURN <ausdr>
Deklaration einer Funktion.

GET Siehe @ ... SAY ... GET.
GO | **GOTO BOTTOM** | **TOP** | <n_ausdr>
Positionieren des Dateizeigers.

IF <bedingung>
<kommandofolge>
{**ELSEIF**
<kommandofolge>...}
[**ELSE**
<kommandofolge>]
ENDIF
Bedingte Anweisung.
INDEX ON <ausdr> **TO** <filename> | (<c_ausdr>) Indizieren der aktiven Datenbank nach dem angegebenen Ausdruck. Es entsteht ein Indexfile (.ndx).
INPUT [<prompt>] **TO** <speichervar> Sequentielle Eingabe von der Tastatur und Zuweisung des eingegebenen Wertes an die Speichervariable.

JOIN WITH <alias> | (<c_ausdr_1>) **TO** <filename> | (<c_ausdr_2>)
FOR <bedingung> [**FIELDS** <feldliste>]
Vereinigen angegebener Felder von zwei Datenbankfiles.

KEYBOARD <c_ausdr> Füllen des internen Tastaturpuffer.

LABEL FORM <filename> | (<c_ausdr_1>) [**SAMPLE**] [<gb>] [**FOR** | **WHILE** <bedingung>]
[**TO PRINT**] [**TO FILE**] <filename>| (<c_ausdr_2>)]
Ausgabe von Briefadressen.
LIST [**OFF**] [<gb>] [<ausdr_liste>] [**FOR** | **WHILE** <bedingung>] [**TO PRINT**]
[**TO FILE** <filename> | (<c_ausdr>)]
Ausgabe der berechneten Werte der Ausdrucksliste.

ISSN 0232-2892
Mikroprozessortechnik, Berlin 4 (1990) 3
S. 65–96

vorgestellt

Multiprozessor-PC Compaq Systempro

Anfang November 1989 stellte Compaq seine beiden ersten Computer mit der Erweiterten Industriestandard-Architektur (EISA) vor. Zum einen als Auftisch-PC den Deskpro 486/25 mit Intels neuem 32-Bit-Prozessor, dem 80486. Zum anderen ein Hochleistungs-Personalcomputersystem für Netz- und Multiuserumgebungen, laut Compaq „das erste PC-System, das Multiuserprocessing ermöglicht". Zum ersten Male biete sich bei einem PC die Möglichkeit, die Leistung des Systems entsprechend den Anforderungen durch den Einsatz eines zweiten, optionalen Systemprozessors zu erhöhen. Die ersten Modelle des Systempro werden noch mit einer 386/33-Prozessorplatine ausgestattet, auf der ein 33-MHz-80386, ein Cachecontroller 82385, 64 KByte Cache und Sockel für die Koprozessoren 80387 und 3167 vorhanden sind. Mit Verfügbarkeit des 80486 kann diese Platine gegen eine 486er ersetzt werden. Ebenso läßt sich als Zweitprozessor anstelle des 80386 dann auch der 80486 verwenden. Mit dem Einsatz eines Zusatz-80386 soll sich die Rechenleistung um bis zu 100 Prozent steigern lassen, bei Verwendung eines Zusatz-80486 auf das Vierfache. Der Einsatz mehrerer Systemprozessoren wird durch Multiprocessingerweiterungen von Netzwerk- und Multiuserbetriebssystemen wie Novell Netware 386, SCO Unix System V/386 und Compaq LAN Manager 386/486 unterstützt. Das Betriebssystem steuert dabei die Zusammenarbeit der beiden Prozessoren und weist ihnen jeweils Anwendungs- und Betriebssystemaufgaben zu. Die vorhandenen Anwendungsprogramme können unverändert weiterverwendet werden, wobei das Multiprocessing für den Nutzer durchschaubar bleibt.
Um höchstmöglichen 32-Bit-Systemdurchsatz zu erreichen – unter Beibehaltung der Kompatibilität –, wurde die Compaq Flexible Advanced Systems Architecture dahingehend erweitert, daß Flex/MP nun auch Multiprocessing und die 32-Bit-EISA-Erweiterungen des ISA-(Industriestandard-)-Busses unterstützt. Der Vorteil liegt darin, daß die Systemprozessoren mit ihren jeweiligen Cachespeichern auch während der Busmasteraktivitäten ununterbrochen arbeiten können. Da Flex/MP getrennte Busse für Prozessor/Arbeitsspeicher und I/O-Peripherie besitzt, können diese Elemente gleichzeitig aktiv sein.
Der bereits genannte EISA-Bus bietet die notwendigen Voraussetzungen für den Einsatz von leistungsfähigen 32-Bit-Controllern, beispielsweise schnellen Netzwerk-Schnittstellencontrollern, Plattencontrollern, Busmaster- und DMA-Controllern. Aufgrund der Kompatibilität zum Industriestandard können jedoch auch weiterhin die vorhandenen und künftigen 8-/16-Bit-ISA-Steckkarten verwendet werden.
Eine Besonderheit ist, daß sich die beiden Systemprozessoren den Hauptspeicher teilen, das heißt gleichzeitig zugreifen können. Durch dieses *Shared System Memory Design* genannte Konzept muß für den zweiten Systemprozessor kein zusätzlicher Arbeitsspeicher installiert werden. Standardmäßig beträgt der 32-Bit-Highspeed Enhanced Page Memory 4 MByte; mit 2-, 8- oder 32-MByte-Modulen kann er bis auf 256 MByte aufgerüstet werden. Die 80-Nanosekunden-DRAMs reduzieren zum einen die Wartezeiten beim Speicherzugriff (wait states) und unterstützen zum anderen Übertragungsgeschwindigkeiten von 33 MBit/s zwischen EISA-Geräten und den Systemprozessoren.
Laut Compaq erstmals bei einem PC angewendet wird als weitere Besonderheit die bisher nur von Großrechnern bekannte Technik des *Drive Array*. Dabei werden mehrere Festplattenlaufwerke als ein einziges Hochleistungslaufwerk angesprochen. Durch die Verteilung der Daten über diese synchronisierten Laufwerke können Datenübertragungen parallel und fast viermal so schnell wie bisher üblich vollzogen werden. Der steuernde IDA-Controller (Intelligent Drive Array Controller) unterstützt bis zu acht interne synchronisierte 210-MByte-Festplatten. Diese interne maximale Festplattenkapazität von 1,68 GByte kann durch den Anschluß von zwei Massenspeichereinheiten extern auf bis zu 4,28 GByte ausgebaut werden.
Besonders in den speicherintensiven Netzwerkumgebungen spielen Fragen der Datensicherheit eine große Rolle, weshalb Compaq dafür mehrere Verfahren vorsieht. Als neue Sicherheitsfunktion werden beim sogenannten *Data Guarding* 25 Prozent der Arraykapazität für die Speicherung der Kopie der Daten in codierter Form verwendet. Daneben unterstützt der Controller auch die *Laufwerksspiegelung*, bei der jeweils die Hälfte der Kapazität einer Festplatte für die Speicherung einer Kopie der Daten in der anderen Hälfte verwendet wird. Schließlich erfüllt das *Controller-Duplexing*, bei dem die Fehlertoleranz mittels eines zweiten Controllers erhöht wird, eine weitere Sicherheitsfunktion.
Abschließend einige weitere Merkmale:
- 11 Steckplätze voller Länge
- Platz für 11 interne Massenspeicher
- integriertes Video Graphics System für VGA, EGA und CGA
- optional Bandlaufwerke (Streamer) bis zu 525 MByte
- ein 5,25-Zoll-Diskettenlaufwerk (1,2 MByte)
- je nach Variante 240 MByte bis zu 840 MByte Festplattenkapazität Standard
- eine parallele, zwei serielle Schnittstellen.

Mit den angegebenen Rechenleistungen von 8 bis 40 MIPS (Millionen Instruktionen pro Sekunde) zielt Compaq auf prozessorintensive Anwendungen wie Datenbankserver in LANs oder als Hostrechner in Multiuserumgebungen. Dies sind Bereiche, in denen bisher Minirechner oder spezialisierte Workstations vorherrschten. Die Firma glaubt, daß mit dem besseren Preis-Leistungs-Verhältnis, den außergewöhnlichen Leistungswerten und Erweiterungsmöglichkeiten des Systempro der Start für eine „vollkommen neue Klasse von Personalcomputern", die als PC-System bezeichnet werden, gegeben ist. Andere Hersteller scheinen dies ähnlich zu sehen. Als Konkurrenten für Multiprozessor-PCs werden demnächst voraussichtlich die taiwanesischen Firmen Mitac und Acer auf dem Markt erscheinen, und auch von IBM wird ein ähnliches PC-System erwartet.

MP-We

ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

MP

Heft 4 · 1990

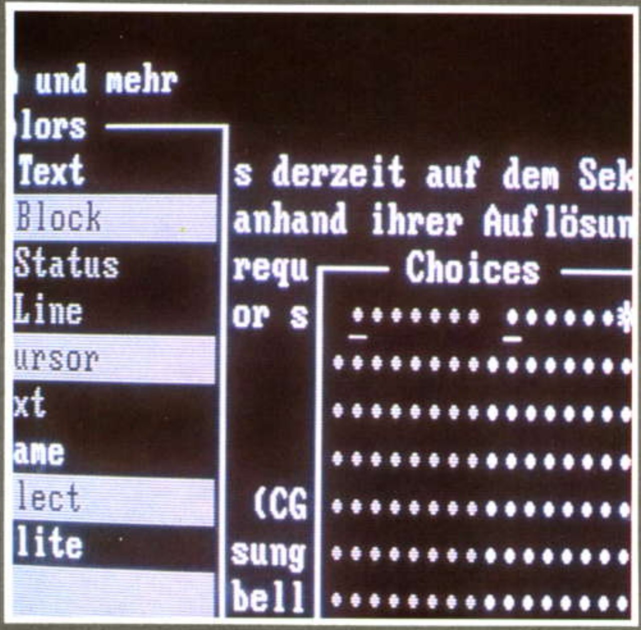
Mikroprozessortechnik

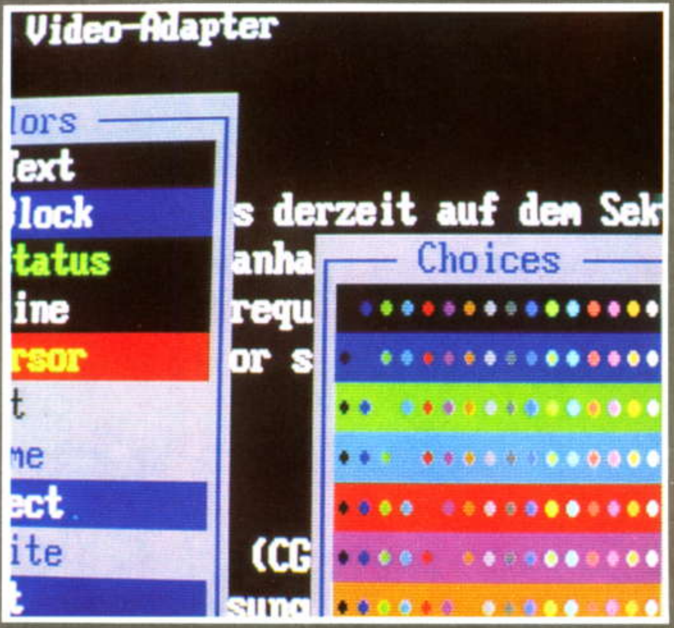
VEB Verlag Technik Berlin
ISSN 0232-2892

mit Beiträgen aus c't

Helfer für den schnellen Zugriff

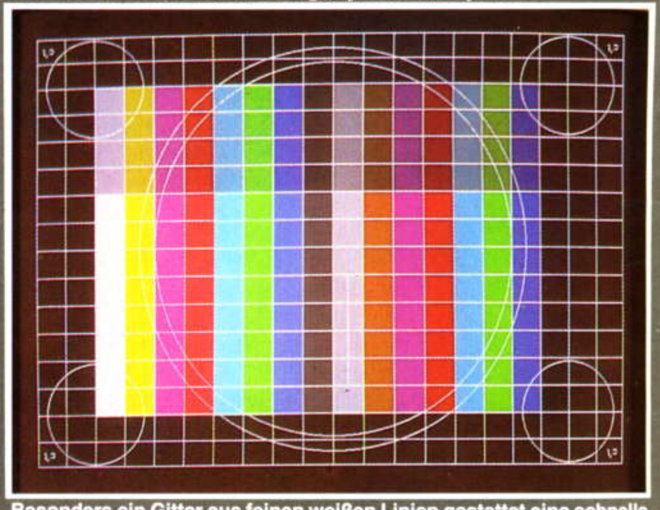
Cache





Grafikadapter-Historie von links nach rechts: Die „groben Zeilen von CGA", daneben dasselbe mit Double-Scan via VGA-Karte, also jede zweite Zeile doppelt dargestellt. Gemeinsames Manko: Die Textzeilen liegen zu dicht übereinander. Weit besser wird es mit Hercules monochrom (hier auf einem Farbschirm wiedergegeben) und EGA. Am feinsten werden die Zeichen im letzten Bild mit Standard-VGA-Auflösung 720 × 400 dargestellt.

Lesen Sie hierzu unseren Beitrag „Systeme mit System"



Besonders ein Gitter aus feinen weißen Linien gestattet eine schnelle und optimale Beurteilung von Monitoren.

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
4. Jahrgang · ISSN 0232–2892

Mikroprozessortechnik, Heft 4 · 90

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag Verlag Technik, Oranienburger Str. 13/14, DDR – 1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Hans Weiß, Verantwortlicher Redakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2870203), Hans-Joachim Hill (Tel. 2870209); Sekretariat Tel. 2870381

Gestaltung Christina Bauer

Titelbild H. J. Eggstein

Beirat Prof. Dr. sc. Thomas Horn, Prof. Dr. Bernd Junghans, Dr. Dietmar Keller, Prof. Dr. sc. Bernd-Georg Münzer, Prof. Dr. sc. Peter Neubert, Prof. Dr. sc. Dr. Michael Roth (Vorsitzender), Dr. Gerhard Schulze, Prof. Dr. sc. Manfred Seifart, Dr. Dieter Simon, Prof. Dr. sc. Dr. Jürgen Zaremba

Lizenz-Nr. 1710 des Presse- und Informationsdienstes der Regierung der DDR

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

Erfüllungsort und Gerichtsstand Berlin-Mitte. Der Verlag behält sich alle Rechte an den von ihm veröffentlichten Aufsätzen und Abbildungen, auch das der Übersetzung in fremde Sprachen, vor. Auszüge, Referate und Besprechungen sind nur mit voller Quellenangabe zulässig.

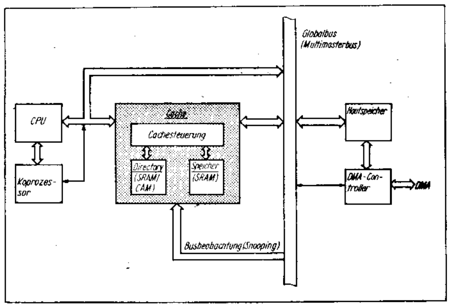
Redaktionsschluß 9. Februar 1990

AN (EDV) 49837

Erscheinungsweise monatlich 1 Heft

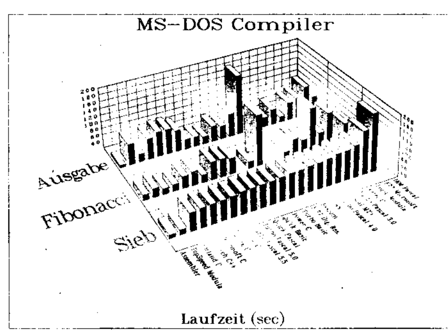
Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *Republik Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Republik Ungarn:* P.K.H.I., Külföldi Előfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nôi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR – 7010 Leipzig und Leipzig Book Service, Talstraße 29, DDR – 7010 Leipzig



Caches (sprich: käschs) sind schnelle Zwischenspeicher, die in Computern zwischen dem Prozessor und dem Hauptspeicher arbeiten. Sie verhelfen dem Prozessor zu einem schnelleren Zugriff auf seine Programme und Daten, so wie die Helfer von Egon Olsen beim Zugriff auf die „Kohle" helfen (siehe Titelbild) – wenn nichts dazwischen kommt. Lesen Sie hierzu unseren Beitrag auf der **Seite 99.**

Im Test:



Nach unseren Testberichten zu CP/M-Programmiersprachen in den Heften 1 und 3 stellen wir Ihnen dieses Mal die verfügbaren Programmiersprachen für MS-DOS vor. Der Beitrag auf der **Seite 103** vergleicht Compiler, Interpreter und Assembler – die Grafik zeigt, daß die schnellsten der Assembler, TopSpeed Modula und Zorland C sind.

In diesem Heft beginnen wir mit einem Einsteiger-Kurs für das gegenwärtig am weitesten verbreitete Betriebssystem für PCs:
MS-DOS der Firma Microsoft. Wir wollen damit dem großen Interesse am Umgang mit der PC-Technik Rechnung tragen.

Vorschau

Für MP 5/1990 bereiten wir für Sie Beiträge zu folgenden Themen vor:
- Mikrocontroller
- Berechnung einer zur Run-Time eingegebenen Funktion
- Residente Programme unter MS-DOS

Inhalt

Gert-Ulrich Vack:
Der Cache 99

Uwe Schulze:
Im Test: Programmiersprachen für MS-DOS 103

Hans-Jürgen Meier:
Ein Mehrprozeßsystem für MS-DOS 105

Michael Schlosser:
Lernmethoden der Künstlichen Intelligenz 106

Detlef Grell, Manfred Bertuch:
Systeme mit System (Teil 2) 108

MP-Kurs 111
Wolfram Schulze, Uwe Schulze:
Einführung in MS-DOS

Frank Schmidt:
Anschluß gewinnen 115

Reinhard Langmann:
Mikroprozessorsteuerungen und grafische Systeme 117

Handelspartner stellen sich vor 120
Soft-Tronik
Mikado

MP-Computer-Club 122
Gärtner:
Zusatzsoftware für Bildungscomputer
Michael Lennartz:
Kennengelernt: Pascal für Kleincomputer
Uwe Schulze:
Undokumentiert: Das DOS-Compatibility-Flag
Bernd Matzke:
Formulardruck mit Nadeldruckern

MP-Börse 124

Entwicklungen und Tendenzen 126

MP-Literatur 128

Clipper-Funktionen 3. US

vorgestellt 4. US
PizazzPlus

Computer-Chaos komplett

Am letzten Wochenende im Februar (24.–25. Februar 1990) fanden sich Computer- und Medienfreaks sowie Fachleute zum „ersten Kommunikationskongreß in der sektorierten Stadt" im Berliner Haus der jungen Talente (HdjT) ein. In der gemeinsamen Veranstaltung des HdjT-Computerclubs und des Chaos Computer Clubs (CCC) aus Hamburg ging es unter dem Motto „Bürgerdiplomatie" um den öffentlichen Erfahrungsaustausch und ums Kennenlernen. Nach anfänglichen Anpassungsproblemen, die sich organisatorisch am Sonnabendvormittag noch bemerkbar machten, hatten die Veranstalter aus Ost und West recht schnell die Kompatibilität hergestellt und das „Chaos geordnet". Für den Besucher stellte sich der Ablauf der Veranstaltungen jedenfalls nicht ungeordneter dar als der anderer Tagungen. Während der Sonnabend vorrangig der technischen Wissensvermittlung vorbehalten war, beispielsweise zu Mailbox-Netzen (übrigens wurde über mehrere Postleitungen Computerkommunikation auch über Landesgrenzen betrieben), ging es am Sonntag vor allem um gesellschaftliche Zusammenhänge und Konsequenzen. Themen waren unter anderem Computerkriminalität, Datenschutz und -sicherheit und Computerviren – jeweils präsentiert von namhaften Referenten aus Ost und West. Die über 400 Besucher der Veranstaltung waren mit dem Gebotenen offensichtlich ebenso zufrieden wie der Leiter des HdjT-Computerclubs, Stefan Seeboldt, und der „Alterspräsident" des CCC, Wau Holland (39), mit dem Ergebnis des Kennenlernens.
Inkompatibilität gab es nur noch bezüglich eines möglichen nächsten Veranstaltungstermins. Seeboldt: „Im Juni ... oder, na ja, schreib' mal Herbst." Holland: „Wann die Ostfreaks wollen, am liebsten im Februar 91 ... oder vielleicht Ostern?" Aber wer erwartet denn heutzutage konkrete Termine, wenn sie über 14 Tage hinausreichen? *MP-We*

Gewendete Börse

So schnellebig ist unsere Zeit: Hatten wir Ihnen in MP 2/90 noch Hinweise zur Rubrik Börse gegeben, die wir als ein Mittel zur Unterstützung der Neuererbewegung ins Leben riefen, so sehen wir uns nun bereits veranlaßt, neuen Gegebenheiten Rechnung zu tragen. Immer mehr Zuschriften erreichen uns mit Angeboten, die nicht der Nachnutzung dienen, sondern unter kommerziellen Gesichtspunkten entstanden sind. Um bei der leider immer noch begrenzten Seitenzahl der MP auch diese Entwicklungen berücksichtigen zu können, werden wir etwa ab Heft 7/90 die Börse auch für kommerzielle Angebote öffnen, indem wir diese für Sie bisher kostenlose Rubrik in Anzeigenseiten umwandeln. Damit bestehen dann für alle Anbieter gleiche Bedingungen.
Zur Vereinfachung des Ablaufes bitten wir Sie, Ihre Börse- bzw. Anzeigenangebote ab sofort an die Anzeigenabteilung des Verlages zu richten; bereits bestätigte Manuskripte werden bis etwa Mitte des Jahres nach dem bisherigen Verfahren veröffentlicht. *Ihre Redaktion MP*

Technologietransfer-Kontakte

Die Zeitschrift *Elektronik-Journal* plant die Einrichtung einer regelmäßigen Technologietransfer-Seite. Dabei wird DDR-Firmen aus den Bereichen Elektronik, Elektrotechnik und aus angrenzenden Gebieten (z. B. Maschinenbau, Meß- und Automatisierungstechnik) ab sofort die Möglichkeit eingeräumt, Kooperationsgesuche direkt an das Elektronik-Journal zu senden. Diese werden dann mit Kennziffer kostenlos veröffentlicht und daraufhin dem Journal zugehende Reaktionen ebenfalls kostenlos an die Firmen in der DDR weitergeleitet.
Mindestens folgende Angaben sollten enthalten sein:
– genauer Kooperationswunsch (Fachgebiet, Dauer usw.)
– Grundinformationen zum Unternehmen einschließlich Adresse, Telefon-Nr. usw.
Ihre Kooperationsgesuche richten Sie bitte an das Elektronik-Journal, Frau Ariane Rüdiger, Europa-Fachpresse-Verlag GmbH, Thomas-Dehler-Str. 27, D-8000 München 83.

Desktop-Dialog kostenlos

Der in der BRD ansässige Desktop-Verlag bietet DDR-Bürgern, die an den elektronischen Druckvorstufen (Desktop Publishing) interessiert sind, für das laufende Jahr den kostenlosen Bezug seiner Zeitschrift Desktop Dialog an. Inbegriffen ist der ebenfalls kostenlose Kennziffern-Service. Nach Ablauf des Jahres wird der weitere reguläre Bezug der Zeitschrift angeboten, über den der Interessent entscheiden kann. Der Verlag erwägt, im nächsten Jahr ein Redaktionsbüro in der DDR zu eröffnen, so daß der Abonnementpreis in Mark der DDR zahlbar wäre. Wer in den Verteiler aufgenommen werden möchte, sende bitte eine kurze Nachricht an folgende Anschrift:
Desktop-Verlag, Postfach 200104, D-6074 Rödermark 2. *MP*

LWL-Übertragungstechnik für 10-MBit/s-Ethernet

Der Lichtwellenleiter (LWL) ist ein zweckgerechtes Medium für die Datenübertragung mit hohen Bitraten. Neben den guten Übertragungsparametern bietet er eine hohe Sicherheit gegen Störeinwirkungen. Um eine reale Möglichkeit für die Nutzung dieses Mediums zu schaffen, entwickelte das ZFT Nachrichtenelektronik in Anpassung an die in der DDR verfügbare Rechentechnik einen optischen Transceiver, der einen Anschluß von LWL über das Transceiver-Interface entsprechend der Ethernet-Spezifikation 2.0 gestattet. Der Transceiver, der auf der Basis einer Zusammenarbeit mit dem VEB Kombinat Robotron entstand, ist modular aufgebaut und gestattet durch den Einsatz unterschiedlicher optoelektronischer Wandler eine Leistungs- und Kostenanpassung an die jeweilige Entfernung. Das Hauptanwendungsgebiet wird in der Verbindung einzelner Rechner oder LANs, die in verschiedenen Betriebsteilen eines städtischen Bereichs installiert sind, gesehen. Im Falle der Verbindung von LANs muß jeweils einer der im Netz eingebundenen Rechner die Funktion des Knotenrechners und über einen zweiten Controller die Verknüpfung mit der Übertragungsstrecke übernehmen.
Unter Berücksichtigung des Zugriffsverfahrens kann für die Punkt-zu-Punkt-Verbindung eine Streckenlänge von bis zu 10 km im allgemeinen als sinnvoll gelten. In diesem Rahmen sind natürlich auch Verbindungen innerhalb ausgedehnter Betriebsgelände möglich. Dabei können schnelle Rechner unter Nutzung eines Sternkopplers auch in einem LAN zusammengeschaltet werden. In diesem Fall darf dann allerdings jede einzelne LWL-Verbindung eine Streckenlänge von maximal 4,5 km nicht überschreiten.
Interessenten für die Übertragungseinrichtung, aber auch andere LWL-Verbindungen für hohe Datenraten, finden im ZFT Nachrichtenelektronik, Außenstelle Dresden, PSF 127, Dresden, 8012 (Tel. 4956012), einen Partner.

Atari-Messe 1990

Vom 24. bis 26. August 1990 wird in Düsseldorf zum vierten Mal die Atari-Messe stattfinden. Nach dem Besucherzustrom von 35000 Interessierten aus dem Vorjahr werden in diesem Jahr weit über 40000 Besucher erwartet. Neben dem umfangreichen Angebot auf den Ständen steht wieder eine Vielzahl von Vorträgen auf dem Programm. Die Themenliste reicht von Schulsoftware über professionelle Branchenlösungen bis hin zu industriellen Computer-Konzepten.
Eine andere Rekordzahl betrifft die Aussteller-Teilnahme: In diesem Jahr werden rund 200 Aussteller das Neueste vom Neuen der Hard- und Softwarepalette rund um den Atari-Computer bieten. Entsprechend ist die Ausstellungsfläche in den Hallen 11 und 12 um ein Drittel größer, das heißt, auf 16000 Quadratmetern ist genügend Platz, um über neueste Produkte und jüngste Entwicklungen zu informieren. Die Atari-Computer-Show ergänzt mit ihrem breiten Angebot an Hard- und dazugehöriger Software die parallel stattfindende Freizeit- und Familienausstellung „Er-Sie-Es aktiv Leben '90". Viele Besucher werden an den Atari-Ständen die Gelegenheit nutzen, Freizeitgestaltung per Computer einmal selbst auszuprobieren.
Wer mit als Aussteller vertreten sein will, kann die Ausstellungsunterlagen anfordern bei: Atari Computer GmbH, Frankfurter Str. 89, D-6096 Raunheim. *MP*

Niederlassung der SE Spezial-Electronic in der DDR

Im Februar eröffnete die Spezial-Electronic KG, Bückeburg, eine Zweigniederlassung in Magdeburg. Sie dient nach den Worten von Günther H. Wuttke, Geschäftsführer der Firma, in erster Linie dazu, Schulungen in Computerbereichen wie Textverarbeitung CAD/CAM-Anwendungen, Betriebssysteme und Computernetzwerke durchzuführen. Zu diesem Zweck wird in der Magdeburger Niederlassung ein Rechnernetzwerk mit etwa 10 Arbeitsstationen, Farbbildschirmen, Druckern und Plottern installiert werden.
Die Adresse der neuen Niederlassung: SPEZIAL ELECTRONIC KG, Otto-von-Guericke-Straße, DDR – 3090 Magdeburg. *MP*

Kostenlose Software-Lizenzen von Borland

Die Borland GmbH bietet allen DDR-Bürgern, die bisher mit kopierter Borland-Software gearbeitet haben, die Möglichkeit, sich bis zum 30. Juni 1990 kostenlos registrieren zu lassen. Neben der Legalisierung der vorhandenen Software bietet sich damit für Entwickler und Anwender in der DDR die Chance, ein günstiges Update auf die neueste Version der jeweiligen Software wahrzunehmen. Die Borland GmbH Deutschland weist ausdrücklich darauf hin, daß sie in keinem Fall DDR-Bürger wegen bisher illegaler Nutzung von Borland-Produkten belangen wird.

Verfahrensweise

- Schreiben Sie einfach an die folgende Adresse und teilen Sie mit, welche Borland-Produkte Sie zur Zeit benutzen. Vergessen Sie bitte nicht, die Versionsnummer anzugeben (z. B. Turbo Pascal 3.0 oder 4.0).

Borland GmbH
z. H. Frau Marion Kappelmaier
Lindwurmstraße 88
D-8000 München 2

- Sie erhalten von Borland daraufhin eine Registrierungsnummer, die Sie als legal lizensierten Anwender der von Ihnen benutzten Software-Kopien ausweist. Sie genießen damit alle Rechte lizensierter Benutzer von Borland-Software.
- Durch die Lizensierung haben Sie zwei Jahre die Gelegenheit, ein Update Ihrer registrierten Software (etwa von Turbo Pascal 5.5) zum üblichen Update-Preis zu erhalten. Dieser liegt in der Regel weit unter der Hälfte des Neuanschaffungspreises. Beispielsweise kostet Turbo Pascal 5.5 neu DM 450,30, das Update von Turbo Pascal 4.0 auf die Version 5.5 knapp unter DM 200,–.

Der Cache

oder Wozu braucht eine CPU einen versteckten Speicher?

Dr. Gert-Ulrich Vack, Dresden

Was ist ein Cache?

Das Wort Cache bedeutet im Englischen *Versteck* oder *ein versteckter Speicher für Proviant.* Bei 16- und 32-Bit-Mikroprozessoren ist zunehmend von Caches die Rede. So gibt es externe und On-chip-Caches, Daten- und Programmcaches, Cache-MMUs (kurz CAMMUs), Adreßübersetzungscaches, Sprungzielcaches, Cachecontroller und sogar Cachehierarchien.

Moderne Mikroprozessoren arbeiten gegenwärtig mit Taktfrequenzen von 33 oder 50 MHz. Beispielsweise hat bereits ein mit 16 MHz getakteter 80386 nach dem Aussenden der Adressen ohne Adreßpipelining nur 77 ns Zeit, die Daten vom Bus zu übernehmen. Ein 33-MHz-80386 benötigt Speicherzugriffszeiten um die 30 ns, damit keine zusätzlichen Wartezyklen eingefügt werden müssen. Leider kann man mit dynamischen RAMs (DRAMs – dynamische Schreib-/Lesespeicher) diese kurzen Zugriffszeiten trotz verschiedener Kunstgriffe (vor allem bezogen auf die Gestaltung der Busschnittstelle) kaum erreichen. Schnelle DRAMs sind gegenwärtig noch teuer und rar, und typisch werden zur Zeit etwa 65 ... 100 ns benötigt, bis nach Übernahme der Adreßinformationen die Daten auf den Leitungen stabil bereitstehen. Spitzenprodukte kommen mit 22 bis 35 ns aus (s. MP 9/89, S. 284).

Mit geeigneten Hardwaremaßnahmen wird nun versucht, den Flaschenhals zwischen Speicher und CPU (zentrale Verarbeitungseinheit) zu erweitern, zum Beispiel durch den Einsatz schneller statischer RAMs (SRAMs). Diese Speicher sind aber ebenfalls teuer und schaltungstechnisch aufwendig (Stromversorgung!). Die erforderlichen Hauptspeicherkapazitäten von mindestens 1 MByte (UNIX benötigt etwa 2 MByte) lassen sich damit ökonomisch nicht erreichen. Perspektivisch könnten einmal BiCMOS-RAMs an Bedeutung gewinnen, bei denen CMOS- und Bipolartechnologien auf einem Chip kombiniert werden. RAMs mit 256 KByte bis 1 MByte in BiCMOS-Technologie beispielsweise erreichen Zugriffszeiten von 5 bis 8 ns. Diese Bauelemente sind aber noch nicht massenhaft verfügbar, und offen bleibt, ob sie dem Geschwindigkeitsboom bei Prozessoren auf Dauer Paroli bieten können. Denn auch in Zukunft ist damit zu rechnen, daß Prozessoren bezüglich der Arbeitsgeschwindigkeiten gegenüber Speichern die Nase vorn haben werden.

Ein wichtiger Ausweg ist deshalb gegenwärtig die Anwendung des Cachekonzepts: Ein schneller SRAM von einigen KByte wird als Zugriffsbeschleuniger zwischen CPU und Speicher geschaltet (Bild 1). Der „Proviant" einer CPU sind die Befehle und die Daten, mit denen sie gefüttert wird. Ganz im Sinne der ursprünglichen Wortbedeutung ist der Cache somit ein gegenüber der Software unsichtbares Versteck für den „CPU-Proviant".

Beim Cachekonzept wird der Umstand ausgenutzt, daß bei der Programmausführung – etwa innerhalb einer Prozedur – meist nur ein kleines Fenster als Ausschnitt aus dem Hauptspeicherinhalt für Code- und Datenzugriffe benötigt wird. Prozeduren in C oder Pascal beispielsweise verbleiben längere Zeit in diesem Speicherbereich (zum Beispiel in DO-Schleifen), ehe wieder auf anderen Adressen stehende Befehlsfolgen oder Daten benötigt werden. Diese Eigenschaft von Programmen wird als *Lokalität* bezeichnet.

Der Cache wird durch einen zugeordneten *Cachecontroller* so gesteuert, daß er die aktuell benötigten Befehle und Daten als Abbild des Hauptspeicherinhalts enthält. So lange der Prozessor die Speicherinhalte bereits im Cache erreicht, kann die CPU ohne Wartezyklen auskommen und mit voller Geschwindigkeit laufen. Wenn Daten oder Befehle nicht im Cache auffindbar sind (z. B. bei Verzweigung in eine andere Prozedur), muß vom Cachcontroller der neue Ausschnitt aus dem Hauptspeicher nachgeladen werden. Diese Situation wird *Cachefehler* (englisch: cache miss) genannt. Sie führt dazu, daß die CPU vorübergehend warten muß, bis der Cache geladen ist.

Der Informationstransfer zwischen Cache und Hauptspeicher erfolgt typisch in Blöcken zu je 2, 4, 8 oder 16 Byte. 32-Bit-CPUs nutzen meist 2- oder 4-Wort-Blöcke (8 oder 16 Byte). Bei einem Cachefehler wird deshalb nicht nur das in diesem Moment gerade benötigte Byte oder Wort geladen, sondern eine „kleine" Speicherumgebung, das heißt noch einige Bytes davor und danach. Mit hoher Wahrscheinlichkeit werden aber auch diese Bytes ohnehin nachfolgend benötigt.

Stehen die benötigten Informationen bereits im Cache, so signalisiert die Auswertelogik des Cachecontrollers einen *Treffer* (englisch: hit), und der Zugriffswunsch des Prozessors kann aus dem Cache-Datenspeicher heraus sehr schnell erfüllt werden. Die Häufigkeit des Auftretens von Treffern bezeichnet man als *Trefferrate* (englisch: hit rate). Typische Werte liegen über 80, meist um die 90 %. Ein „langsamer" Hauptspeicher, der nur mit Wartezyklen betreibbar wäre, kann dadurch mit einem Cache in einem (praktisch nicht erreichbaren) 0,1- bis 0,2-Wartezyklenmodus genutzt werden. Caches sind derzeit bei Prozessortaktfrequenzen ab 20 MHz und Speichergrößen ab 1 MByte ökonomischer als schnelle Hauptspeicher.

Wie ist ein Cache strukturiert und wie wird er verwaltet?

Caches kann man nach verschiedenen Prinzipien strukturieren und verwalten (Bild 2). Unabhängig von der konkreten Variante enthält jeder Cache einen SRAM-Datenspeicherteil mit m Blockpuffern – den *Cachezeilen* – und einen Kennzeichnungs- und Adressierungsteil mit m Adreßeinträgen – das Cache*directory.* Man unterscheidet

- voll assoziative Caches (englisch: full associative cache)
- n-Wege-satzassoziative Caches (englisch: n-way set associative cache) und
- direktabbildende Caches (englisch: direct mapped cache).

Für das Adressierungsschema eines 32-Bit-Prozessors (4 GByte Hauptspeicher/32 Adreßbits) werden diese Varianten nachfolgend vorgestellt. Der Cache-Datenspeicher soll 2 KByte in 128 Zeilen (m = 128) aufnehmen können (16 Byte je Zeile).

Beim **voll assoziativen Cache** (Bild 2a) kann ein Hauptspeicherausschnitt in jeder beliebigen der 128 Zeilen stehen. Beim Speicherzugriff durch den Prozessor muß der Controller möglichst unverzögert und des-

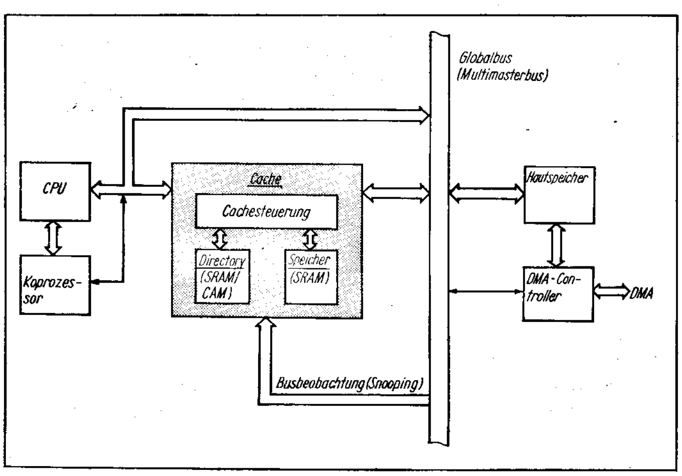


Bild 1 Einbettung eines Caches in ein Multimastersystem

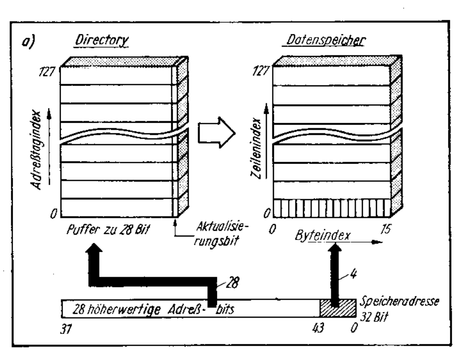


Bild 2 Verwaltungsprinzipien von Caches a) voll assoziativer Cache (ohne LRU-Bits)

halb für alle Zeilen gleichzeitig prüfen, ob eine Zeile die adressierten Speicherzellen enthält. Diesen Adreßvergleich übernimmt die *Trefferlogik* (englisch: hit logic). Die Adreßspeicherzellen für das Cachedirektory bilden gemeinsam mit der Trefferlogik einen *inhaltsadressierten Speicher CAM* (englisch: content adressable memory). CAMs sind Spezialspeicher, die universell auch für andere Zwecke einsetzbar sind und zunehmend als Einzelschaltkreise zur Verfügung stehen. Die parallele Trefferlogik eines voll assoziativen Caches ist schaltungstechnisch

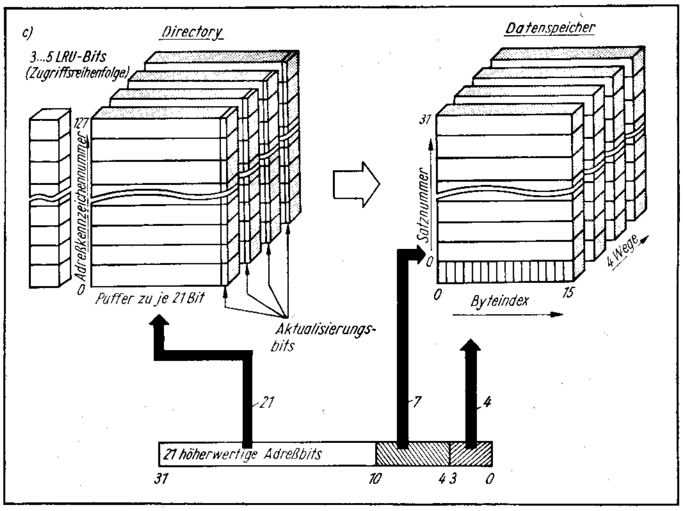


Bild 2
Verwaltungsprinzipien von Caches
c) n-Wege-satzassoziativer Cache

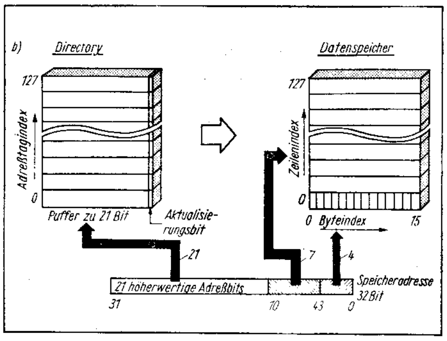


Bild 2 Verwaltungsprinzipien von Caches
b) direktabbildender Cache

sehr aufwendig, so daß dieses Prinzip praktisch kaum angewendet wird, zumal sich mit einfacheren Organisationsformen ähnlich hohe Trefferraten erreichen lassen.

Beim **direktabbildenden Cache** (Bild 2b) kann jeder Hauptspeicherausschnitt nur in einer bestimmten Zeile stehen. Zur Zeilenadressierung in unserem Beispiel dienen 7 Bit der vom Prozessor ausgesendeten Adresse als Index. Mit den Adreßbits 0 bis 3 werden die 16 einzelnen Bytes in der Cachezeile adressiert. Die Adreßbits 4 bis 10 bilden den 7-Bit-Index für die Zeilenadressierung (Wertebereich 0 bis 127). Die Zeile Nr. 3 (Zählung ab 0, Index 3) kann so Informationen ab den Speicheradressen xxxx xxxx xxxx xxxx xxxx x000 0011 0000 (x steht für beliebige Belegungen mit 0 oder 1) enthalten. Zur Unterscheidung von Speicherausschnitten mit gleichem Index ist zusätzlich die Auswertung der Adreßbits 11 bis 31 notwendig. Diese Bits kennzeichnen unterschiedliche Adreßbereiche und werden deshalb als *Kennzeichen* (englisch: tag, wörtlich Etikett) bezeichnet und im Cachedirectory im *Kennzeichenfeld* (englisch: tag field) als Adreßinformation eingetragen. Beim Prozessorzugriff auf ein Speicherbyte wird aus den Adreßbits 4 bis 10 der Index gebildet und geprüft, ob das damit indizierte Kennzeichenfeld den korrekten höherwertigen Adreßteil enthält. Bei Übereinstimmung liegt ein Treffer vor, und in der zugeordneten Zeile steht der gewünschte Speicherausschnitt. Anderenfalls muß wieder nachgeladen werden. Die Auswertelogik eines auf diese Weise organisierten Caches ist einfacher und deshalb billiger zu gestalten als beim vollassoziativen Cache, da nur jeweils genau ein Eintrag ausgewertet werden muß. Caches mit dieser Organisationsform werden deshalb besonders häufig angewendet.

Eine Zwischenstellung zwischen beiden Organisationsprinzipien nimmt der **n-Wege-satzassoziative Cache** ein (Bild 2c). Die 128 Cachezeilen unseres Beispiels werden zum Beispiel in 4 Wegen zusammengefaßt. Jeweils 4 Zeilen gleichen Niveaus bilden einen *Satz* (englisch: set). Der Beispielcache hätte somit folgende Organisation: 4 Wege, je Weg 32 Zeilen, je Zeile 16 Byte. Die Sätze eines n-Wege-satzassoziativen Caches werden wie ein direkt abbildender Cache verwaltet und die Zeilen eines Satzes wie ein assoziativer Cache. Diese Kombination der eingangs genannten Verfahren stellt einen ökonomischen Kompromiß zwischen Aufwand und Leistungsfähigkeit dar. Eine vom Prozessor gesuchte Information kann nun in einer von 4 Zeilen stehen. Man benötigt neben der Indizierlogik, die jetzt den Satzzugriff vermittelt, zusätzlich eine parallele Auswertelogik, die im genannten Beispiel jedoch nur 4 Adreßvergleiche gewährleisten muß – nämlich für jeweils 4 Zeilen eines Satzes. Caches mit dieser Organisationsform sind ebenfalls recht häufig, meistens ist n aus Aufwandsgründen auf 2 oder 4 beschränkt, selten auch n = 8. Der direktabbildende Cache ist als 1-Weg-Cache interpretierbar.

Wie werden Cachedaten ersetzt und aktualisiert?

Die Cacheverwaltung ermöglicht zwei wichtige Strategien: die Strategie zur Aktualisierung des Hauptspeichers, wenn vom Prozessor in den Cache geschrieben wird, und die Ersetzungsstrategie bei n-Wege-satzassoziativen Caches, nach der die n Zeilen eines Satzes verwendet werden.

Das Problem der *Cacheaktualisierung* erwächst aus dem Erfordernis, den Hauptspeicherinhalt dem Cacheinhalt bei Schreibzugriffen nachzuführen. Wenn zum Beispiel die Werte von globalen Variablen während der Programmabarbeitung im Cache geändert werden, so müssen die neuen Werte auch in den Hauptspeicher übernommen werden. Die Übereinstimmung der Dateninhalte von Cache und Hauptspeicher bezeichnet man als *Cachekohärenz*. Zwei Verfahren sind zur Gewährleistung der Kohärenz üblich: copy back (manchmal auch als write back bezeichnet) und write through.

Beim **copy back** (Zurückschreiben) wird der Hauptspeicherinhalt erst dann geändert, wenn eine Cachezeile mit neuen Informationen belegt werden soll und der ursprüngliche Dateninhalt modifiziert wurde. Das bereits vorgestellte Kennzeichenfeld enthält dazu für jede Zeile eine Aktualisierungsinformation, das *Gültigkeitsbit* (englisch: valid bit oder manchmal auch dirty bit genannt). Bevor die Zeile mit neuen Daten vom Hauptspeicher belegt wird, schreibt die Cachesteuerung den alten Inhalt zum Hauptspeicher zurück, falls das Aktualisierungsbit eine Änderung des Dateninhalts signalisiert.

Beim **write through** (ständig schreiben) hingegen werden Schreibzugriffe in den Cache sofort auch an den Hauptspeicher weitergeleitet, so daß die Kohärenz permanent gesichert ist. Damit vergrößert sich zwar die Busbelastung (wichtig in Multiprozessorsystemen) – und der Prozessor wird „gebremst" –, zugleich ergeben sich aber übersichtlichere Verhältnisse für den Hauptspeicherinhalt.

Eine Modifikation dieses Verfahrens ist *buffered write through* (gepuffertes ständiges Schreiben), bei dem die Hauptspeicherschreibzugriffe in zusätzlichen Auffangregistern, den Schreibpuffern, zwischengespeichert werden. Der Prozessor erhält bereits bei Übernahme der Adreßinformation und der Daten durch den Cache die Fertigmeldung, so daß er das Ende der Schreiboperation zum Hauptspeicher nicht abwarten muß. Erst wenn der Hauptspeicherbus frei ist, werden die Daten aus den Auffangregistern zurückgeschrieben. Nur in den seltenen Fällen, wenn auch die Schreibpuffer gefüllt sind, muß die CPU warten.

Bei Assoziativecaches muß als zweites wichtiges Operationsprinzip die **Ersetzungsstrategie** genannt werden, nach der neu einzulesende Hauptspeicherausschnitte den Zeilen eines Satzes zugeordnet werden. Besonders wichtig ist das Verfahren *least recently used* (kurz LRU, am längsten nicht benutzt). Dabei wird als nächstes der Puffer beschrieben, auf den am längsten nicht vom Prozessor zugegriffen wurde.

Zur Markierung der LRU-Zugriffsreihenfolge muß bei einem n-Wege-Assoziativcache eine zusätzliche Steuerinformation im Kennzeichenfeld verwaltet werden (LRU-Bits). Die Länge dieser Steuerinformation in Bit ist der aufgerundete duale Logarithmus von n!, für einen 4-Wege-Cache also bereits 5 Bit. Wenn man von der notwendigen Länge dieser Informationen Bits einspart, spricht man von Pseudo-LRU. Dabei wird aber in Kauf genommen, daß nicht immer auf die exakt am längsten nicht verwendete Cachezeile zugegriffen wird.

Caches und DAMs, Multiprocessing und Coprocessing

Die Cachekohärenz ist auch in all den Fällen besonders zu beachten, in denen mehrere Prozessoren oder Controller (Master) gemeinsamen Zugriff auf den Hauptspeicher erhalten.

① DMA-Betrieb. Der DMA-Controller wird meist an die Verbindung zwischen Cache und Hauptspeicher geschaltet (Bild 1). Da der Prozessor längere Zeit nur auf den Cache zugreift, kann der DMA-Controller gleichzeitig ungestört mit dem Hauptspeicher Daten austauschen, ohne den Cache zu belasten.

② In Multiprozessorsystemen kommunizieren die Prozessoren im geteilten Zugriff über den Globalspeicher. Wenn ein Prozessor zum Beispiel eine Nachricht von einem anderen Prozessor über den Globalspeicher in den Cache übernommen hat und diese Nachricht anschließend im Globalspeicher vom Sender erneut aktualisiert wird, muß auch im betreffenden Cache eine Datenaktualisierung entsprechend dem neuen Hauptspeicherinhalt vorgenommen werden.

③ Auch Koprozessoren können Hauptspeicherzugriff erhalten. Meist sind diese jedoch als Slave direkt an den Wirtsprozessor angekoppelt (z. B. in der Konfiguration 80386/80387) und können den Hauptspeicher nur über den Wirtsprozessor erreichen.

Cachekohärenz muß also meist in beiden Richtungen gesichert werden. Ein einfaches Verfahren ist die Verwendung von nicht „cachebarem" Speicher. Durch geeignete Chipselect- und Adreßkombinationen wird gewährleistet, daß gemeinsam von mehreren Busmastern genutzte Hauptspeicherbereiche nicht in den Cache kopiert werden können, sondern nur im Direktzugriff erreichbar sind. Damit entfällt das Kohärenzproblem, allerdings auf Kosten der Zugriffsgeschwindigkeit, der Flexibilität und zusätzlichen Hardwareaufwands.

Leistungsfähige Cachecontroller (z. B. der Cachecontroller 82385 oder die Cache-Memory-Management-Unit des Clipper-Prozessors) ermöglichen deshalb eine besondere *Busüberwachung* (englisch: snooping). Hauptspeicherzugriffe durch andere Busmaster werden über einen Adreßvergleich mit den Adressen der im Cache gehaltenen Speicherausschnitte überwacht. Wenn andere Busmaster in den Hauptspeicher auf Adressen schreiben, die in einem Cache enthalten sind, sorgt die Cachesteuerung selbständig für eine Aktualisierung des Cacheinhalts aus dem Hauptspeicher. Umgekehrt werden im Cache aktualisierte Daten bei Lesezugriffen von anderen Busmastern auf den Hauptspeicher vor der Ausführung der Leseoperation in den Hauptspeicher zurückgeschrieben.

Caches werden in Multimastersystemen mit virtueller Speichertechnik meist zwischen Speicherverwaltungseinheit und Speicher, also auf der Seite der physischen Adressen, angeordnet. Das kommt der Sicherung der Cachekohärenz entgegen, da somit für alle Master übereinstimmend dieselben virtuellen Adreßzuordnungen von Speichersegmenten gelten. Vor allem in Multiprozessorsystemen ist man vor Kohärenzproblemen jedoch nicht völlig gefeit. Als *trashing* („alles demolieren") bezeichnet man die Situation, wenn Cachezeilen permanent durch verschiedene Prozesse umgeladen werden, also praktisch zwischen zwei oder mehr Adreßbereichen pendeln.

Beim *cache-flushing* (Cache ausräumen) wird der Cacheinhalt in Konfliktsituationen zwangsweise ungültig gemacht, wobei aber zuvor meist der Hauptspeicher mit geänderten Cachedaten aktualisiert wird. Anschließend muß der Cache neu vom Hauptspeicher geladen werden. Cache-flushing kann in On-chip-Caches von 32-Bit-Prozessoren zum Beispiel zeilenweise ausgelöst werden.

Cachedimensionierung

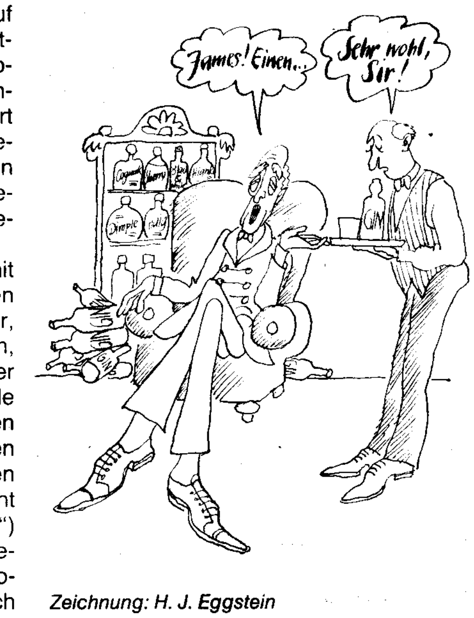
Zur Dimensionierung von Caches gibt es mehrere Freiheitsgrade. Diese betreffen, ausgehend von der verfügbaren SRAM-Größe,

- die Zeilenlänge (Byte je Zeile); daraus ergibt sich die Anzahl der Zeilen im Cache.
- das Assoziativitätsmaß (Anzahl der Wege); daraus ergibt sich die Anzahl der Zeilen je Weg und die Anzahl der Zeilen je Satz.
- die Aktualisierungs- und die Ersetzungsstrategie.

Kritisch bei der Cachedimensionierung ist die Festlegung der Zeilenlänge. Zu lange Zeilen führen dazu, daß im Cache weniger Speicherausschnitte gespeichert werden können. Deshalb kann ein Cachefehler schneller auftreten. Das Nachladen langer Zeilen dauert länger und erfordert eine schnellere Busschnittstelle, damit der Prozessor nicht zu lange warten muß. Auch werden bei zu langen Zeilen mehr überflüssige Bytes gelesen, die unter Umständen in keinem oder nur in einem schwachen inhaltlichen Bezug zum aktuellen Programmkontext stehen.

Im Rahmen des Architekturentwurfs ermittelt man per Simulation vor dem Schaltkreisentwurf geeignete Cachevarianten, die der geforderten Trefferhäufigkeit entsprechen (üblich sind 80 ... 90 %) und die mit angemessenem Aufwand verwirklicht werden können. Als Randbedingungen sind die Arbeitsgeschwindigkeiten der Trefferlogik und die für die SRAMs erforderlichen Chipflächen (für Cachedirectory, Datenspeicher) sowie die Signalverzögerungen zu berücksichtigen.

Bereits auf der Ebene der Architektursimulation benötigt man also für bestimmte Schaltungsdetails eines Caches Realisierungsinformationen, um geeignete Architekturfestlegungen treffen zu können. Beispielsweise bestand beim Entwurf der Trefferlogik des Adreßübersetzungscaches des 80386 die Forderung, die Trefferauswertung in weniger als 11 ns abzuschließen (20 MHz Taktfrequenz). Daraus ergaben sich spezifische Anforderungen an die Wahl einer geeigneten Schaltungstechnik. Beim Entwurf des RISC-Prozessors MIPS wurde ebenfalls in enger Verbindung mit der späteren schaltungstechnischen Umsetzung die Cachegröße optimiert. Der Architekturentwurf wurde hierbei bezüglich der SRAM-Größe dadurch begrenzt, daß die Datenwege auf dem Chip nicht zu lang sein durften.



Zeichnung: H. J. Eggstein

Von wesentlichem Einfluß auf die Trefferrate ist der Charakter der Software. Die Trefferraten können so durchaus zwischen 50 und 90 % differieren. Als Eingabedaten für die Cachesimulation verwendet man häufig Programme und Daten der Software, die künftig mit der CPU-Cache-Kombination abzuarbeiten ist. Das kann beispielsweise ein kompletter C-Compiler sein.

In enger Verbindung zum Schaltkreisentwurf stehen auch die Strukturierung in Zeilen und das Assoziativitätsmaß: Je höher assoziativ ein Cache organisiert ist, desto größere Treiberleistungen sind für die Trefferlogik notwendig.

Aus Sicht einer schnellen Programmausführung ist eine hohe Trefferrate anzustreben. Grundsätzlich gilt, daß die Zeilenlänge proportional zur SRAM-Größe gewählt werden sollte. Bei Cachegrößen bis 1 KByte sind die Trefferraten voll assoziativer Caches und direkt abbildender Caches etwa gleichwertig. Oberhalb 1 KByte ergeben sich geringfügige Vorteile voll assoziativer Caches von einigen Prozent.

Der Schaltungsaufwand für das Cachedirectory ist nicht zu vernachlässigen. Beim bereits diskutierten 4-Wege-Cache ist jedem Satz ein Kennzeichenfeld zuzuordnen, das im konkreten Fall bei voller LRU-Ersetzungsstrategie 5 LRU-Bits enthalten muß. Dazu kommen noch 4 Gültigkeitsbits für die 4 Zeilen jedes Satzes. Außerdem müssen je Zeile 21 Bit für die Adreßkennzeichen berücksichtigt werden.

Neben dem Datenspeicher-SRAM, der die Nutzinformationen enthält, ist also auch der Speicher für das Cachdirectory mit allen Adreß- und Zusatzinformationen als SRAM auszulegen. Auf diesen Speicher greift die Cachesteuerung zu. Im Unterschied zum Datenspeicher-SRAM, der sehr regulär aufgebaut ist, muß dieser Directory-SRAM verschiedenste Hardwaredirektzugänge zu einzelnen Bits gestatten. Der Chipflächenbedarf ist dafür recht erheblich: Er beträgt beispielsweise 80 % beim Cache-Controller 82385 (dieser ist für die direkt abbildende Caches und für 2-Wege-Assoziativcaches mit externen SRAMs ausgelegt). 28000 Bit sehr schneller SRAM wurden benötigt.

Wo werden Caches angewendet? Zum Beispiel im 80486 ...

Caches werden vor allem als Speicherzugriffsbeschleuniger eingesetzt. Sie können als externe Caches oder als On-chip-Caches ausgelegt sein. Die Leistungsfähigkeit der Busschnittstelle (Übertragungsbandbreite) sowie die technologischen Möglichkeiten, vor allem der mögliche Integrationsgrad, sind entscheidende Kriterien für die Variantenauswahl. Ein Beispiel für einen Cachecontroller in Form eines Einzelschaltkreises für den Aufbau externer Caches von maximal

32 KByte mit direktabbildender oder 2-Wege-Assoziativorganisation ist Intels 82385.
Prinzipiell ist auch eine Trennung der Cachefunktionen für Programmspeicher und für Datenspeicher möglich. Dieses Konzept wird zum Beispiel in Verbindung mit Harvardarchitekturen genutzt, bei denen getrennt Bussysteme für Programmspeicher und für Datenspeicher existieren (Beispiele hierfür sind die Prozessoren Clipper und Motorola 88000). Wenn man die Caches auf dem Chip trennt, wird die Harvardarchitektur gewissermaßen auf dem Chip nachgebildet. In bezug auf die Entwicklung des 80860 zeigte sich, daß es günstiger sein kann, diesen Weg zu beschreiten und besser im Interesse einer großen Busbandbreite zum Speicher einen 64-Bit-Datenbus vorzusehen als getrennte Bussysteme für Programm- und Datenspeicher. An diesem Beispiel wird der enge Zusammenhang verschiedener Architekurentwurfsentscheidungen besonders deutlich.
Üblich sind weiterhin Kombinationen aus Cachecontroller und Speicherverwaltungseinheit MMU (z. B. Motorola 88000 oder Clipper). Bei besonderen Geschwindigkeitsanforderungen verwendet man mehrstufige Caches. Sind On-chip-Caches zu klein, kann man extern Zusatzcaches einführen (dieser Weg wird beim 80860 für spezielle Grafikanwendungen notwendig werden). Ultraschnelle Galliumarsenid-Prozessoren werden bereits mit Taktfrequenzen um 200 MHz betrieben. Die erste Cachestufe arbeitet beispielsweise mit GaAs-Speichern, die folgenden Stufen können langsamere Speicher höherer Komplexität enthalten. Neben dieser Hauptanwendung von Caches gibt es noch andere Einsatzfälle für das Cacheprinzip:
Der *Adreßübersetzungscache* (englisch: translation lookaside buffer, TLB) ist logisch parallel zur Adreßübersetzungseinheit eines Prozessors geschaltet. Er enthält on-chip die zuletzt verwendeten Zuordnungspaare virtueller zu physischen Adressen einer Speicherverwaltung. Für alle im TLB enthaltenen Referenzen kann somit auf die (die Programmausführung bremsende) Adreßübersetzung verzichtet werden.
In RISC-Prozessoren werden *Sprungzielcaches* (englisch: branch target cache) eingesetzt, um die Parallelarbeit (Pipelining) der einzelnen Prozessorteile bei auftretenden Programmverzweigungen nicht allzusehr ins Stocken geraten zu lassen. Bei der Ankopplung von Massenspeichern an einen Rechner werden große *Datencaches* verwendet, um die Arbeitsgeschwindigkeiten von Massenspeicher und Hauptspeicher einander anpassen zu können.
Cacheprinzipien können mit anderen Speicherorganisationsformen kombiniert werden. Vor allem in RISC-Prozessoren findet man Stack-Caches und Registerstack-Caches. Dabei wird die Zeilenorganisation an die vom Prozessor verarbeiteten Datenformate direkt angepaßt (z. B. 16 oder 32 Bit), wodurch sich eine effektive Nutzung seitens der Software ergibt.
In Verbindung mit einem effektiven Cachekonzept verfügen moderne 32-Bit-Prozessoren über Blocktransfermodi, bei denen mehrere zusammenhängende Bytes mit nur einem Bustakt transferiert werden können (Burstmodus, transaktionsorientierte Busschnittstelle). Vor allem bei On-chip-Caches lassen sich damit die Busbandbreite und somit die Geschwindigkeit des Cacheumladens steigern. Weitere spezielle Maßnahmen sind zum Beispiel software- oder hardwaregesteuertes Vorausladen (englisch: prefetch) des Caches, um der Erhöhung der Programmausführungszeit bei Cachefehlern zu begegnen. Optimierungskriterien in bezug auf die Software und die Auslegung der Busschnittstelle sind auch die Trennung oder Vereinigung von Daten- und Programmcaches und die Proportionen zwischen beiden.
Schließlich ist es auch möglich, Cacheorganisationsformen in der Software nachzubilden. Software-Caches werden beispielsweise in manchen Forth-Systemen im Softwareinterface zum Massenspeicher oder als Teil des Wörterbuchs (Wortcache) für die zuletzt verwendeten Wortreferenzen eingesetzt.
Abschließend soll als Beispiel die Nutzung von Caches in einem System mit dem 80486 skizziert werden. Die CPU enthält auf dem Chip zwei getrennte Caches: einen 8-KByte-Cache, der für Daten und Befehle gemeinsam genutzt wird und einen Adreßübersetzungscache (vergleiche hierzu MP 9/1989, Seite 278).
Der Programm-/Datencache ist 4-Wege-satzassoziativ organisiert. Eine Zeile enthält 16 Byte. Als Ersetzungsstrategie wurde Pseudo-LRU gewählt. 3 LRU-Bits treffen eine zweistufige 1-aus-2-Entscheidung über die am längsten nicht genutzte Zeile eines Satzes (Bit 0 hält die Zeilen 1/2 und 3/4 auseinander, Bit 1 unterscheidet dann zwischen Zeile 1 und 2 und entsprechend Bit 2 zwischen Zeile 3 und 4). Die Sätze des Caches werden mit den Adreßbits 4 bis 10 angesprochen. Das Adreßkennzeichen enthält die Bits 11 bis 31. Je Satz gibt es 4 Gültigkeitsbits. Die Trefferrate kann bei über 90 % liegen (Streubereich für verschiedene Programme 75 bis 97 %).
Die transaktionsorientierte Busschnittstelle gestattet bei einer Taktfrequenz von 33 MHz 32-Bit-Blockübertragungen zwischen Cache und Hauptspeicher mit 105,8 MByte je s (für eine Zeile werden 5 Takte benötigt). Als Aktualisierungsstrategie wurde buffered write through gewählt. Die 4 Schreibpuffer sind jeweils 32 Bit breit und intern als Ringpuffer (FIFO, first in – first out) organisiert.
Der Adreßübersetzungscache ist ebenfalls als 4-Wege-Assoziativcache mit modifiziertem LRU-Algorithmus ausgelegt (Zeilenlänge 20 Bit). Er kann die 32 letzten Adreßübersetzungen speichern. Bei Cachetreffern können damit die sonst notwendigen mindestens 4 Takte für die Adreßübersetzung eingespart werden.
Außerdem gibt es auch im 80486 den bereits aus dem 80386 bekannten, in der Software unsichtbaren Segmentdeskriptorcache, der parallel zu den 6 sichtbaren Segmentregistern verwaltet wird. Bei Segmentwechsel werden automatisch die Segmentdeskriptorinformationen (Segmentbasisadresse, Segmentlimit und weitere Segmentattribute) in das dazugehörige 8-Byte-Cacheregister übernommen. Alle Speicherzugriffe innerhalb des aktivierten Segments werden nun über das Segmentregister an das Cacheregister weitergeleitet. Betriebssystemfunktionen (z. B. zur Überwachung von Zugriffsrechten und Segmentbereichsüberschreitungen) brauchen so die notwendigen Informationen nicht permanent aus dem im Hauptspeicher abgelegten Deskriptorblock auszulesen, sondern können direkt die on-chip gespeicherten Informationen auswerten. On-Chip-Caches können über die Software in ihrem Verhalten steuerbar sein. Im 80486 gibt es spezielle Befehle wie INVD (englisch: invalidate – invalidieren, ungültig machen), die den Cache leeren. Eine modifizierte Version von INVD ist WBINVD (write back invalidate), dabei wird zuvor der Hauptspeicher aktualisiert. Der Cache läßt sich mit einem Enablebit freigeben oder sperren, und einzelnen Seiten des Hauptspeichers können über das Seitendirectroy zusätzliche Informationen für die Cachesteuerung zugeordnet werden. Im 80486 sind außerdem bereits (in weiser Voraussicht) Steuerbits vorgesehen, mit denen später einmal eine Umschaltung auf verschiedene Aktualisierungsstrategien möglich sein wird.

TERMINE

1. Berliner Unix-Forum

WER? Unix-Entwickler- und -Anwendergemeinschaft (UNIX-EAG)
WANN? 22. Mai 1990, 9.00 bis 18.00 Uhr
WO? Berliner Kongreßhalle
WAS?

- Jahrestagung der UNIX-EAG
- Unix-Ausstellung gemeinsam mit der UniWare Computer GmbH (Berlin-West); sie bietet vor allem Firmen aus der BRD die Gelegenheit, ihre Hard- und Software zu präsentieren.

WIE? Interessenten wenden sich bitte an: VEB Leitzentrum für Anwendungsforschung Berlin – Softwarebetrieb CAD/CAM, Bereich F13, Jacques-Duclos-Straße 47–52, Berlin, 1156; Tel. 37 80 34 80

Grützbach

Fachtagung „MIDOS 16 Bit"

WER? Arbeitsgemeinschaften (B) der KDT für Information und Dokumentation der Bezirke Potsdam und Berlin
WANN? 17. Mai 1990
WO? Potsdam, Kulturhaus „Hans Marchwitza"
WAS? Anwendererfahrungen und Ausblick auf die Weiterentwicklung des Programmsystems MIDOS für die Nutzung in der wissenschaftlich-technischen Information und in Bibliotheken
WIE? Teilnahmewünsche richten Sie bitte an: Kammer der Technik, Bezirksvorstand Potsdam, SB Wissenschaft und Technik, Weinbergstraße 20, PSF 27, Potsdam, 1561; Tel. 2 34 26 oder 2 40 62

Winkler

Im Test:

Programmiersprachen für MS-DOS

Uwe Schulze, Berlin

Nachdem wir in MP 1 und 3/90 Testberichte zu Programmiersprachen, die unter dem Betriebssystem CP/M laufen, veröffentlichten, vergleichen wir in diesem Beitrag Programmiersprachen, die unter MS-DOS verfügbar sind.

Für MS-DOS steht eine Vielzahl von Compilern und Interpretern zur Verfügung. Damit die zu messenden Laufzeiten der Compiler in einer leicht auswertbaren Größenordnung liegen, wurden für das Primzahlensieb des Erathostenes 100 Iterationen bis 8190 ausgeführt (statt 10 bei den CP/M-Compilern) und die Fibonacci-Funktion zehnmal bis 22 protokolliert. Es ist also gegenüber den CP/M-Compilern der Faktor 10 anzusetzen. Die Compilezeiten beziehen sich auf die Erzeugung eines EXE-Files (bei Turbo 3.0: COM-File) auf Festplatte. Manche Compiler bieten auch die Möglichkeit der Compilierung im Speicher (Turbo-Pascal) oder eines Syntax-Checks ohne Codeerzeugung (Quick-C). Vorteilhaft zum Schreiben portierbarer Quellen sind Optionen zum Erzwingen des jeweils genormten Sprachstandards (Turbo-C).

In einer Zeit, in der Produktnamen ohne Präfixe wie Turbo oder Power nicht auszukommen scheinen, mag ein TopSpeed-Modula kein besonderes Aufsehen erregen. Doch der Name hält, was er verspricht. So kann sich die Einschätzung „nett, aber langsam" von Turbo-Pascal-Bauer Anders Hejlsberg über das Werk seines ehemaligen Team-Kollegen Niels Jensen wohl kaum auf das Laufzeitverhalten beziehen, sondern allenfalls auf die Compilezeit – und das auch nur im Vergleich mit Borland-Produkten (Tafel 1). Die erzeugten Programme verdienen zu Recht das Prädikat „handcodiert". Man darf auf TopSpeed-C und die angekündigten Möglichkeiten der Mixed-Language-Programmierung gespannt sein.

Daß Microsoft-C schnellen Code erzeugt, ist gemeinhin bekannt; auch wird erstmals umfassender OS/2-Support geboten. Bemerkenswert ist aber, daß der kürzeste Code von einem Pascal-Compiler kommt: Auch bei den Übersetzungszeiten und im Entwicklungskomfort liegt Turbo-Pascal vorn. Interessant auch zu entdecken, daß es für Verbesserungen Grenzen gibt: In der Tat benötigt Turbo-Pascal 5.0 für die Fibonacci-Funktion – bei unbestrittenen Vorteilen in den anderen Leistungsparametern – mehr Zeit als die Vorgängerversion.

Beeindruckend ist die Laufzeit des C 88-Compilers für die Fibonacci-Funktion. Selbst in Assembler geht es nicht schneller (jedenfalls nicht bei mir). Gleichzeitig zeigt das aber, daß Vorsicht bei der Verallgemeinerung von Aussagen über das Laufzeitverhalten geboten ist. Beim Primzahlensieb rangiert derselbe Compiler unter „ferner liefen".

Neben dem Test einiger häufig verwendeter Programmkonstrukte (Speicherzugriffe und Schleifen beim Sieb, Unterprogrammrufe, Parameterübergaben und Möglichkeiten der Rekursion bei Fibonacci) liegt der Wert dieser Benchmarks wohl vor allem in der allgemeinen Akzeptanz und der Möglichkeit des Vergleichs /1/, /2/, /3/. Während Dateiarbeit aufgrund der gemeinsamen Nutzung der BDOS-Funktionen keine signifikanten Unterschiede erwarten läßt, wäre ein ausführlicher Arithmetiktest sicher auch interessant.

Die bei C-Compilern verbreitete Möglichkeit der Optimierung brachte nur auf dem P8000 deutlich bessere Ergebnisse (siehe Tafel 2). Hier ist auf Kosten der Compilezeit eine gleichzeitige Optimierung von Codegröße und Laufzeit möglich. Die untersuchten C-Compiler für MS-DOS erlauben wahlweise nur die Optimierung nach einem Zielkriterium (außer Quick-C).

Quick-C wird ab Version 2.0 nicht mehr als Zugabe zum „großen" Microsoft-C-Compiler, sondern als selbständiges Produkt vertrieben. Gegenüber der Vorgängerversion wurden die erzeugten Programme wieder etwas schneller, allerdings auf Kosten von Übersetzungszeit und Codegröße. Geliefert wird neben einer separaten Kommandozeilenversion des Compilers jetzt auch ein Quick-Assembler – der erste Assembler in integrierter Entwicklungsumgebung.

Eine exponierte Stellung unter den C-Compilern nimmt Zortech C++ ein. Dieser Compiler verarbeitet neben Standard-C- und Assembler(!)-Quellen den objektorientierten Sprachstandard nach Stroustrup /4/. Und das nicht als Präcompiler – erzeugt werden direkt ausführbare EXE-Files.

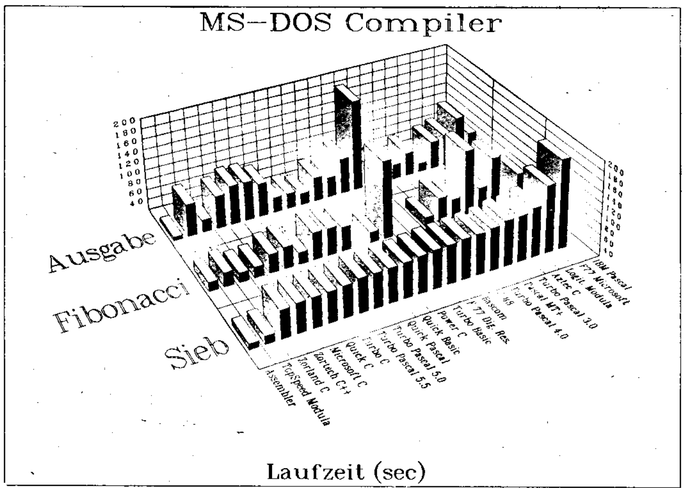
Tafel 1 Testergebnisse auf einem EC 1834 (4,7 MHz) nach Laufzeit des Primzahlensiebes geordnet

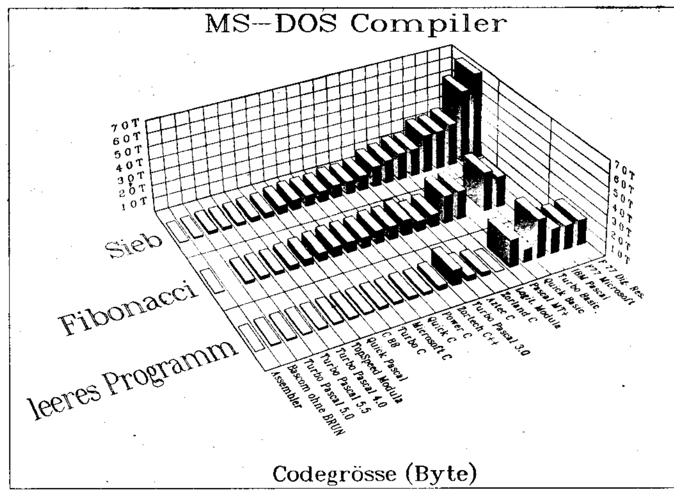
Compiler	Primzahlensieb (100 Iterationen)			Fibonacci-Funktion (zehnmal bis 22)			Bildschirm-ausgabe (32 000 Zeichen)	leeres Programm	
	Laufzeit in min	Compile-zeit in min	Code-größe in Byte	Laufzeit in min	Compile-zeit in min	Code-größe in Byte	Laufzeit in min	Compile-zeit in min	Code-größe in Byte
JPI TopSpeed Modula-2	0:36	0:13	3687	0:47	0:13	3651	1:23	0:09	1611
Zorland C Version 2.05	1:13	0:22	17888	0:38	0:28	9632	0:45	0:18	2880
Zortech C++ Version 1.07	1:16	0:17	9929	0:41	0:16	9891	1:24	0:12	3500
Microsoft-C 5.1	1:17	0:52	7843	0:56	0:52	7805	1:36	0:42	2311
Quick-C 2.0	1:18	0:22	8136	1:02	0:25	8098	1:35	0:17	2404
Turbo-C 2.0	1:20	0:12	7602	0:43	0:11	7532	1:25	0:08	2310
Turbo-Pascal 5.5	1:25	0:02	2496	1:13	0:02	2544	1:18 (CRT) 0:50	0:01	1312
Turbo-Pascal 5.0	1:25	0:02	2496	1:13	0:02	2544	1:18 (CRT) 0:50	0:01	1312
Quick-Pascal 1.0	1:26	0:04	3904	1:07	0:04	4800	1:18 (CRT) 0:23	0:03	1760
Quick-Basic 4.50	1:27	0:36	29284	–	–	–	1:28	0:28	10240
Power C Version 1.1.6	1:30	0:36	9840	0:41	0:34	9808	1:01	0:22	2096
Turbo-Basic	1:33	0:02	29616	2:50	0:02	29593	1:31	0:01	28992
Fortran 77 Digital Research Version 4.1	1:34	1:12	67456	–	–	–	3:17	0:44	18816
Bascom 1.0	1:37	0:07	1412 + BRUN	–	–	–	1:04	0:06	1181
C 88 Version 2.3	1:38	0:15	6144	0:38	0:14	6144	1:00	0:12	1536
Turbo-Pascal 4.0	1:41	0:02	2736	1:08	0:02	2656	1:18 (CRT) 0:50	0:01	1360
Pascal MT+	1:47	0:46	20992	0:58	0:46	20992	1:07	0:44	20480
Turbo-Pascal 3.0	1:48	0:02	11716	1:26	0:02	11652	0:38	0:01	11433
Aztec-C Version 1.05j	1:55	0:26	16064	1:05	0:23	7776	1:15	0:10	3824
Logitech-Modula *)	2:29	0:31	20800	1:57	0:26	20649	1:22	0:14	112
Fortran 77 Microsoft Version 3.20	3:11	0:52	56288	–	–	–	1:50	0:46	18218
IBM-Pascal Version 1.0	9:00	0:56	32660	1:11	0:54	23340	1:12	0:41	18648

*) keine EXE-Dateien, sondern LOD-Files, die mit „M2 Dateiname" abgearbeitet werden

```
MODULE Sieve;
FROM IO IMPORT WrLn, WrInt, WrStr;
CONST Size=8190;
VAR Flags: ARRAY[0..Size] OF BOOLEAN;
    i, prime, k ,count, iter: CARDINAL;
BEGIN
  WrLn; WrStr("100 iterations");
  FOR iter:= 1 TO 100 DO
      count:=0;
      FOR i:=0 TO Size DO Flags[i]:= TRUE
      END;
      FOR i:=0 TO Size DO
          IF Flags[i] THEN
             prime:=i*2+3;
             k:=i+prime;
             WHILE k <= Size DO
                   Flags[k]:=FALSE;
                   INC(k,prime);
             END;
             INC(count);
          END;
      END;
  END;
WrLn; WrInt(count,6); WrStr(' primes');
END Sieve.
```

Bild 1 Primzahlensieb in Modula-2 (JPI)





Tafel 2 ... und in C auf einem P8000 (4 MHz)

Compiler	Primzahlensieb (1000 Iterationen)			Fibonacci-Funktion (zehnmal bis 22)		
	Laufzeit in min	Compile-zeit in min	Code-größe in Byte	Laufzeit in min	Compile-zeit in min	Code-größe in Byte
C Unix-P8000	0:65	0:30	6886	0:22	0:27	6774
optimiert	0:33	0:51	6876	0:20	0:43	6762

Tafel 3 ... und einige Interpreter auf einem EC 1834

Sprache	8086-Forth	GW-Basic	RUN/C-Interpreter
Primzahlensieb, Laufzeit für 1 Iteration in min	0:04	2:40	34:30

Tafel 4 Zum Vergleich: Assembler

Assembler	Primzahlensieb (100 Iterationen)			Fibonacci-Funktion (zehnmal bis 22)			Bildschirmausgabe (3200 Zeichen)
	Laufzeit in min	Übersetzungszeit in min	Codegröße in Byte	Laufzeit in min	Übersetzungszeit in min	Codegröße in Byte	Laufzeit in min
MASM 5.0 oder TASM 1.0	0:31	0:11	672 (EXE) 159 (COM)	0:38	0:08	683 (EXE) 427 (COM)	0:28 (BIOS) 0:53 (BDOS)

Tafel 5 Die gleichen Benchmarks auf einem A 7100 (4,7 MHz)

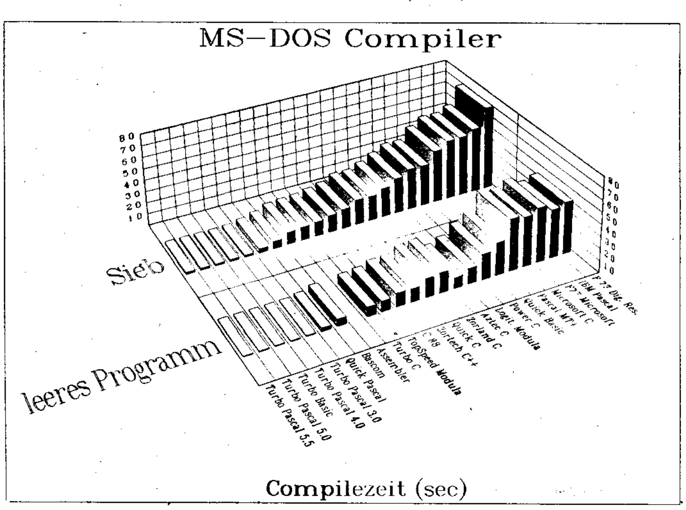
Compiler	Primzahlensieb (100 Iterationen)			Fibonacci-Funktion (zehnmal bis 22)			Bildschirm-ausgabe (3200 Zeichen)	leeres Programm	
	Laufzeit in min	Compile-zeit in min	Code-größe	Laufzeit in min	Compile-zeit in min	Code-größe	Laufzeit in min	Compile-zeit in min	Code-größe
Turbo-Pascal 3.0 unter SCP 1700	2:29	0:09	12 KByte (83 records)	1:47	0:07	12 KByte (82 (records)	0:55	0:07	12 KByte (81 (records)

Der noch etwas schnellere Zorland-Compiler stammt übrigens vom selben Autor – Walter Bright.

Einige Compiler legen ihre Datenbereiche statisch im EXE-File an. Augenfällig wird das durch die Differenz von etwa 8000 Byte zwischen SIEVE.EXE und FIBO.EXE bei Atztec-C und IBM-Pascal. Die Untersuchung zeigt, daß sich bei der Verkleinerung des Feldes für die Primzahlenberechnung die Filegröße äquivalent ändert.

Als State of the Art gelten heute Compiler mit integrierter Entwicklungsumgebung (vor allem Editor), kontextsensitivem Hilfssystem und Sourcecode-Debugger. Wegbereiter sind die Turbo-Compiler von Borland und die Quick-Compiler von Microsoft. Das bisher fehlende Quick-Pascal (mit objektorientierten Features wie Turbo-Pascal 5.5) ist seit kurzem verfügbar.

Um einen inkrementellen Compiler – wie der Hersteller vorgibt – handelt es sich bei Quick-Pascal zwar nicht (das bedeutete Codeerzeugung parallel zum Editieren), aber die Eingabe von Schlüsselwörtern, Zeichenketten und Kommentaren wird ständig überwacht und farblich verschieden dargestellt. Interessant ist der direkte Vergleich zwischen Turbo- und Quick-Pascal; beide Compiler bieten den gleichen Sprachumfang bis Turbo 5.0 (Quellfiles können problemlos ausgetauscht werden) und vergleichbare objektorientierte Erweiterungen.



Tafel 6 Laufzeiten für Turbo 5.0 (ohne Optimierung) auf verschiedenen Prozessoren (1/100 Sekunde Genauigkeit)

Prozessor	Taktfrequenz in MHz	Laufzeit Sieb in min	Laufzeit Fibonacci in min
I 8088	4,7	2:04,79	1:45,51
V 20	4,7	2:03,89	1:38,20
I 8086	4,7	1:26,66	1:12,80
I 8088	8,0	1:16,07	1:02,78
V 20	8,0	1:14,48	0:59,93
I 8086	8,0	0:50,42	0:46,97
I 80286	8,0	0:24,50	0:25,16
I 80286	12,0	0:16,70	0:19,61
I 80386	20,0	0:07,69	0:09,23
I 80386	25,0	0:05,60	0:09,83

Literatur

/1/ Strauß, D.: Wettstreit zweier C-Compiler. mc, München 8 (1989) 1, S. 112
/2/ Wollschläger, P.: Topspeed-Modula – Turbo statt Tuning. Computer persönlich, Haar (1989) 4, S. 97
/3/ Zerbe, K.: C für Profis. C't, Hannover (1988) 7, S. 58
/4/ Stroustrup, B.: Die C++ Programmiersprache. Addison Wesley 1987

Ein Mehrprozeßsystem für MS-DOS

Hans-Jürgen Meier, Dresden

Die Gründe für die Arbeiten an einem weiteren Mehrprozeß-System (Multitask-System) für MS-DOS sind sehr vielgestaltig. Im Vordergrund stand zu keiner Zeit die Schaffung eines vergleichbaren Produktes wie beispielsweise Concurrent DOS XM, Wending-DOS und anderen in /1/ /2/ genannten Systemen, sondern das Sammeln von Erfahrungen beim Entwurf und der Implementierung eines erweiterungsfähigen Mehrprozeß-Systems für die Lehre und Forschung. Von besonderem Interesse waren dabei die Probleme bei der Erweiterung eines bereits existierenden Einprozeß-Betriebssystems um die Mehrprozeß-Komponente. Von Anfang an wurde auf eine nachnutzungsfähige Form orientiert, die für eine breite Anwendung geeignet ist. Die folgenden Gesichtspunkte wurden zugrunde gelegt:

- Im Zuge der weiteren Vernetzung dezentraler Rechentechnik werden für effektive Lösungen quasi parallellaufende Prozesse beispielsweise für Spool- und Serveraufgaben benötigt.
- Der Einsatz von PCs als Leitrechner für Steuerungsaufgaben erfordert ein Mehrprozeß-System für langsame Echtzeitforderungen. Solche Systeme existieren auch schon, verfügen jedoch nicht über die gleichen komfortablen Programmierumgebungen für Programmiersprachen, Datenbanken und anderes wie MS-DOS.
- Die Möglichkeit des Entwurfs und der Implementierung von prozeßorientierter Software und deren quasiparalleler Abarbeitung auf dem PC muß gegeben sein.

Es wurde das bereits existierende Einprozeß-Betriebssystem MS-DOS aufgrund seiner Aktualität und Verbreitung ausgewählt.

MS-DOS gilt heute als Quasi-Standard-Betriebssystem für IBM-kompatible PCs. Vom Hersteller Microsoft wurde MS-DOS bisher nur als ein Einnutzer-Einprozeß-Betriebssystem (Single User Single Task) vertrieben. Obwohl eine ganze Reihe von Ansätzen zu erkennen sind, wie beispielsweise der Aufbau der Driver, gibt es noch keine entsprechende Mehrprozeß-Version. Diese Lücke schließen inzwischen eine ganze Reihe von MS-DOS kompatible Mehrprozeß-Betriebssysteme anderer Hersteller, von denen ein Teil in /1/ einer Analyse unterzogen wurde. Sie laufen zwar alle auf einem zum IBM PC/XT kompatiblen Computern, können aber ihre vollen Möglichkeiten erst auf PCs der AT-Klasse entfalten.

Ein wesentlicher Aspekt bei der Wahl eines Betriebssystems ist der Zweck, für den es eingesetzt werden soll. Das Betriebssystem MTS (Multi Task System) wurde als ein zu MS-DOS kompatibles Mehrprozeß-Betriebssystem entwickelt. Durch eine weitestgehende Hardwareunabhängigkeit läuft es auf dem A 7150, dem EC 1834 und allen anderen IBM-kompatiblen PC der MS-DOS Version 3.3. Die angestrebte MS-DOS-Kompatibilität legt auch das Einsatzgebiet von MTS fest. Ziel war die Schaffung eines Mehrprozeßsystems, das eine Erweiterung des Dienstumfanges von MS-DOS darstellt. Der Einsatz von MTS in eingebetteten 16-Bit-Rechnern als Prozeßsteuerkern ist nicht vorgesehen, kann aber durch eine entsprechende Abrüstung erreicht werden.

Die MS-DOS-Dokumentation spricht von Prozessen, obwohl es nur Prozeduren sind. Wegen der Konsistenz mit der Originalliteratur wird deshalb der Begriff Prozeß auch für die ursprünglichen MS-DOS-Prozeduren verwendet. Die neuen prozeßspezifischen Dienste werden auf MS-DOS aufgesetzt. Somit existiert eine scheinbare Parallelität des Prozeßsystems von MS-DOS und dem neuen Prozeßverwaltungssystems (PVS) von MTS. Das MS-DOS-Prozeßsystem kann dabei als ein Subsystem betrachtet werden. Aus diesem Grund ist es möglich, die prozedurale MS-DOS-Dienstschnittstelle beizubehalten, wodurch die weitestgehende Nutzung der für MS-DOS verfügbaren Software möglich ist.

Prozesse können dynamisch eingerichtet und wieder gestrichen werden. Bei der Erzeugung eines Prozesses wird vorausgesetzt, daß der erforderliche Programmkode von einem externen Direktzugriffsdatenträger ladbar ist. Bei der Nutzung von Systemressourcen, von Synchronisationsmitteln und von Ereignissen werden verbrauchbare und nicht verbrauchbare Betriebsmittel unterschieden. Verbrauchbare Betriebsmittel sind Signale, Nachrichten und die Zeit. Nicht verbrauchbare Betriebsmittel sind der Hauptspeicher, Files, Ein-/Ausgabe-Geräte und der Prozessor. Für das Prozessorscheduling sind folgende Strategien wählbar:

- feste Prioritäten, bei gleichen Prioritäten wird nach dem FIFO-Prinzip (First In First Out) ausgewählt
- Zeitscheibenverfahren, wahlweise mit und ohne Prioritäten.

Bei der Übermittlung von Signalen oder Nachrichten kommt damit das No-wait-send-Prinzip zur Anwendung. Ein Prozeß, der eine Nachricht abgibt, kann somit in jedem Fall weiter ausgeführt werden, auch wenn noch kein Prozeß auf diese Nachricht wartet. Ein anfordernder Prozeß muß gegebenenfalls auf eine Nachricht warten. Die Anwendung der Semaphore liegt in der Hand der Programmautoren, er muß die Eindeutigkeit der verwendeten Semaphore sichern. Dies betrifft beispielsweise die Doppelbelegung. Analoges gilt aber auch für die Kooperation von Prozessen. Sie müssen sich bei der Auftragserfüllung ihrer Wechselwirkung *bewußt* sein.

In diesem Zusammenhang muß darauf hingewiesen werden, daß MTS vorrangig für neue Anwendungssoftware geschaffen wurde, die prozeßorientiert arbeitet. Denn nur damit können die Möglichkeiten von MTS voll genutzt werden. Denkbar wäre beispielsweise die Integration eines MS-DOS-Rechners in ein Steuerungssystem. Die anfallenden Daten können so in einem allgemein zugänglichen Dateiformat gespeichert werden und mit Hilfe von existierender Systemsoftware auf dem MS-DOS-Rechner weiterverarbeitet werden. Natürlich kann der Benutzer auch mit mehreren Standardprogrammen wie MASM und dBase III Plus parallel arbeiten. Bei mehreren größeren Programmsystemen wird jedoch schnell die Grenze des Hauptspeichers erreicht. Für ein einwandfreies Arbeiten des Mehrprozeßsystems müssen sich alle verwendeten Programme an die MS-DOS-Dienstschnittstelle halten.

Die neuen Dienste des PVS werden über den Interrupt 21H unter Angabe eines entsprechenden Kodes aufgerufen. In definierten Registern sind die Eingangsparameter bereitzustellen bzw. werden die Ausgangsparameter geliefert. Die Parameterübergabe und -übernahme wurde der MS-DOS-Konvention angepaßt. Dem Anwender wird eine Unterstützung in den Programmierumgebungen Assembler und C angeboten. Dabei stehen ihm auf Assemblerniveau definierte Makros für den Dienstaufruf zur Verfügung. In C werden ihm die Dienste als Funktionen bereitgestellt. Neben der MS-DOS-Dienstschnittstelle wird durch COMMAND.COM auch das Kommunikationsprotokoll zum Benutzer beibehalten und entsprechend den neuen Möglichkeiten erweitert. Dem Benutzer werden Kommandos angeboten, die ihn über die Prozeßzustände, die Größe der Warteschlangen und die Speicherbelegung informieren. Weiterhin existieren Kommandos zur Prozeßmanipulation, beispielsweise zum Streichen eines Prozesses oder zur Änderung der Priorität eines Prozesses.

Literatur

/1/ Daus, J.: Jenseits von MS-DOS, PC Magazin (1988) 7, S. 30

/2/ Joepgen, H. G.: PC Mode für Concurrent CP/M. MC (1984) 11, S. 111

KONTAKT

VEB ROBOTRON Vertrieb Dresden, Leningrader Straße 9, Dresden, 8010, Tel. 4870

Computermarkt DDR

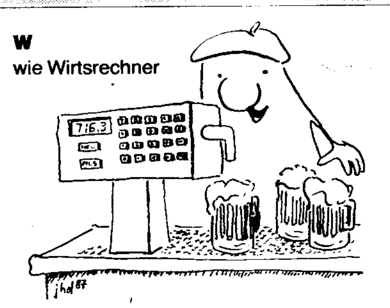
Nach der lange verordneten Enthaltsamkeit bezüglich internationaler Hard- und Software überschlagen sich jetzt Kooperationen von DDR-Betrieben mit großen internationalen Firmen. So war auf der Leipziger Frühjahrsmesse 1990 zu erfahren, daß mehrere Produkte von CA Computer Associates (deren bekanntestes das legendäre Supercalc sein dürfte) in Zukunft durch das Leitzentrum für Anwendungsforschung (LfA) vertrieben werden sollen. Nähere Informationen erhalten Sie unter der Anschrift Jacques-Duclos-Straße 47–52, DDR–1156 Berlin; Tel. 37803610. Die CeBIT 90 brachte eine weitere Neuigkeit: Der südkoreanische Computerhersteller LGI und die Ashton-Tate GmbH (bekannt durch das weit verbreitete Datenbanksystem dBase) beabsichtigen, auf Initiative der Fa. Interlloyd KuFAG über den RFT-Industrievertrieb exclusive Hard- und Software anzubieten.

Nähere Informationen dazu erhalten Sie beim Industrievertrieb Rundfunk und Fernsehen, Bausestraße 6, DDR–7010 Leipzig; Tel. 474211.

Kleines Lexikon der Mikrorechentechnik

W

wie Wirtsrechner



Zeichnung: Jens-Helge Dahmen

Lernmethoden der Künstlichen Intelligenz

Michael Schlosser, Karl-Marx-Stadt

Michael Schlosser (34) studierte von 1972 bis 1976 an der Technischen Hochschule Karl-Marx-Stadt Mathematik. Anschließend arbeitete er als Systemprogrammierer im VEB Datenverarbeitungszentrum Karl-Marx-Stadt. Seit 1985 ist Michael Schlosser als wissenschaftlicher Assistent an der Sektion Informatik der heutigen Technischen Universität Karl-Marx-Stadt auf dem Gebiet des Wissenserwerbs in der Künstlichen Intelligenz tätig.

Bei der klassischen Computernutzung ist es erforderlich, zur Lösung einer vom Nutzer gestellten Aufgabe einen vollständigen und korrekten Algorithmus aufzustellen und diesen in einer für den Computer verständlichen Sprache zu formulieren. Solche Programme sind relativ aufwendig in ihrer Erzeugung und nur für ein ziemlich eingeschränktes Aufgabenspektrum anwendbar. Ein vorgegebenes Programm erzeugt (aus meist notwendigen Eingabeinformationen) die Lösung eines Problems (siehe Bild 1).

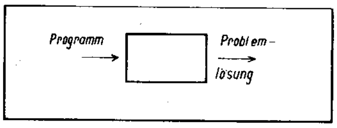


Bild 1
Computernutzung (ohne Lernkomponente)

Mit dem weiteren Fortschreiten von Hard- und Software ist man bestrebt, den Menschen von der relativ langwierigen Tätigkeit des Programmierens zu entlasten und diese auf die Seite des Computers zu verlagern, dem Computer also „kreative Fähigkeiten" und „intelligentes Verhalten" zu vermitteln. Lernen, eines der grundlegenden Merkmale intelligenten Verhaltens, ist definiert als eine erfahrungsbedingte Modifikation des Verhaltens /1/. Dieser Lernbegriff gilt sowohl für tierisches als auch für menschliches Lernen. Lernprozesse schließen ein:
– den Erwerb von neuem Wissen
– die Einordnung von neuem Wissen in bereits existierendes
– eine effektive Darstellung des Wissens
– die Entwicklung von kognitiven (auf Erkenntnis beruhenden) und motorischen Fähigkeiten durch Instruieren oder Erfahrungen
– die Entdeckung neuer Theorien durch Beobachtungen und Experimente.
Eine Übertragung der Fähigkeit des Lernens auf den Computer stellt Bild 2 (als Ergänzung zu Bild 1) dar.

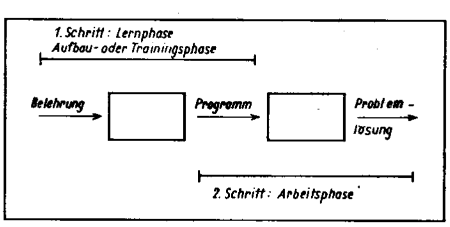


Bild 2
Computernutzung (mit Lernkomponente)

In der Lernphase wird zunächst ein Programm (im weitesten Sinne) „erlernt", wobei häufig auch Rückkopplungen auftreten (z. B. beim Lernen aus Beispielen). Anschließend wird in der Arbeitsphase das „erlernte" Programm (wie auch bereits in Bild 1) zur Lösung eines gestellten Problems benutzt.
Das maschinelle Lernen als noch ziemlich junge Disziplin – erste Ansätze entstanden Ende der fünfziger Jahre – ist heute zwar noch weit davon entfernt, wirklich lernende Computersysteme zu schaffen, doch sind weltweite Aktivitäten zu einer immer weiteren Annäherung an dieses Ziel im Gange. Die Entwicklung und Vervollkommnung maschineller Lernmethoden ist wesentlicher Forschungsschwerpunkt innerhalb der Künstlichen Intelligenz. Das liegt nicht zuletzt an der Notwendigkeit automatischen Wissenserwerbs.
Das Gebiet des maschinellen Lernens umfaßt gegenwärtig drei grundlegende Forschungsrichtungen:

– *theoretische Forschung:* theoretische Erforschung möglicher Lernmethoden, unabhängig von einem Anwendungsgebiet
– *anwendungsorientierte Forschung:* Entwicklung und Analyse von Lernmethoden in eng abgegrenzten Anwendungsgebieten
– *kognitive Simulation:* Erforschung und Computersimulation menschlicher Denkprozesse

Die aufgeschriebene Reihenfolge soll dabei keine Wertung darstellen. Alle drei Richtungen sind notwendig für die weitere Entwicklung des maschinellen Lernens als Wissenschaftsdisziplin innerhalb der Künstlichen Intelligenz. Es bestehen sehr enge Verflechtungen zwischen den genannten Richtungen, was etwas näher erläutert werden soll. Indem viel Forschungsaufwand primär in eine dieser Richtungen gesteckt wird, führt der Erkenntnisgewinn hier häufig auch zu Fortschritten in einer der beiden anderen Richtungen. So können sich theoretische Untersuchungen besonders positiv auf die Entwicklung von Lernmethoden in einem speziellen Anwendungsgebiet auswirken. Weiterhin tragen gut entwickelte praktische Lernmethoden wesentlich zur theoretischen Fundierung bei. In ähnlicher Weise kann die Untersuchung menschlicher Denkprozesse zur Theorie und Praxis des maschinellen Lernens beitragen und umgekehrt.
Im grundlegenden und wohl bisher umfangreichsten Werk zum maschinellen Lernen /2/ werden folgende drei Klassifikationen maschineller Lernsysteme vorgeschlagen:
① Klassifikation auf der Basis der benutzten Lernstrategien (Die Lernstrategien selbst sind geordnet nach dem Umfang an Inferenzen, die das System über die verfügbaren Informationen durchführt.)
② Klassifikation auf der Basis der Wissensdarstellung
③ Klassifikation entsprechend dem Anwendungsgebiet.
Jeder Punkt des Raumes, der durch die drei angegebenen Dimensionen definiert ist, entspricht einer speziellen Lernstrategie, die eine besondere Wissensdarstellung in einem speziellen Anwendungsgebiet benutzt.
Im folgenden werden erforschte Werte entlang der ersten Dimension detaillierter beschrieben; Werte entlang der beiden letzten Dimensionen werden nur erwähnt. Künftige Forschungen können neue (bislang noch nicht untersuchte) Punkte innerhalb des Raumes möglicher Lernmethoden darstellen. Der große Raum der Lernmethoden ist bisher erst wenig erforscht. Existierende Lernsysteme entsprechen nur einem kleinen Teil des gesamten Raumes, weil sie nur eine kleine Zahl möglicher Kombinationen der Werte darstellen.

Klassifikation auf der Basis der benutzten Lernstrategie

Die Lernstrategien sollen nach dem Umfang an Inferenzen (das sind Ableitungen, Schlußfolgerungen), die ein Lernsystem selbständig über die verfügbaren Informationen ausführt, unterschieden werden. Das Wissen eines Systems steigt durch eigene Inferenzen an. Dabei betrachten wir folgende zwei Extreme:
– keine Inferenzen
– „beträchtlicher" Umfang an Inferenzen (beim gegenwärtigen Entwicklungsstand soll dies den maximal möglichen Umfang an Inferenzen bedeuten).
Zwischen diesen beiden Extremen bewegen sich alle möglichen Lernsysteme. Die Einteilung in Tafel 1 wurde /2/ entnommen.

Tafel 1 Einteilung der Lernsysteme nach der benutzten Lernstrategie

Lernstrategie	
1. Auswendiglernen (rote learning)	
2. Lernen durch Instruieren, (learning form instruction, learning by being told)	Zunahme des Umfangs an Inferenzen
3. Lernen durch Analogie (learning by analogy)	≙
4. Lernen aus Beispielen (learning from examples)	Abnahme des Gehalts der Lerninputs
5. Lernen durch Beobachten und Experimentieren (learning from observation and discovery, unsupervised learning)	

Auswendiglernen

Auswendiglernen erfordert keinerlei Inferenzen oder andere Transformationen des Wissens auf seiten des Lernenden. Dem Lernenden werden alle Informationen explizit vorgegeben; die sonst jedem Lernprozeß eigene Kreativität fehlt in diesem Fall völlig. Das Auswendiglernen ist nur der Vollständigkeit halber in dieser Übersicht aufgenommen worden; es handelt sich um keine Lernmethode im eigentlichen Sinne. Varianten dieser Methode enthalten:
– Lernen durch Programmierung (entspricht der herkömmlichen Programmierung eines Computers, siehe Bild 1)
– Lernen durch Erinnerung an gegebene Fakten (kann z. B. durch Anwendung eines klassischen Datenbanksystems erfolgen).

Lernen durch Instruieren

Lernen durch Instruieren entspricht dem Wissenserwerb durch einen Lehrer oder aus einer anderen Quelle, zum Beispiel einem Lehrbuch. Dabei ist es erforderlich, daß der

Lernende das Wissen aus der Eingabesprache in die intern benutzte Darstellungsart umformt und daß neue Informationen in das bisher vorhandene Wissen integriert werden. Folglich muß der Lernende einige Inferenzen selbständig ausführen, aber der größte Teil der Arbeit verbleibt beim Lehrer, der das Wissen in einer geeigneten Weise zur Verfügung stellen muß.

Lernen durch Instruieren ist vergleichbar mit üblichen Ausbildungsmethoden. Die Aufgabe des maschinellen Lernens besteht darin, ein Lernsystem zu schaffen, das Instruktionen oder Ratschläge akzeptiert und das erworbene Wissen effektiv speichern und anwenden kann.

Als ein Beispiel des Lernens durch Instruieren soll das Programm F00 /3/ dienen, mit dessen Hilfe (problemabhängige) Suchalgorithmen in einen vorhandenen (problemunabhängigen) Suchalgorithmus eingebaut werden können. Eine gegebene Aufgabenstellung (z. B. das Acht-Damen-Problem) soll durch vollständige Suche gelöst werden. Folgender Algorithmus sei dem System zu Beginn bereits bekannt:

Ausgehend von einem Anfangszustand (leeres Brett) werden nacheinander acht Felder mit Damen belegt. Anschließend wird geprüft, ob die aktuelle Belegung eine Lösung des Problems darstellt (keine zwei Damen befinden sich auf einer Waagerechten, Senkrechten oder Diagonalen). Bei positivem Ergebnis wird die aktuelle Belegung als Lösung ausgegeben, andernfalls verworfen und weitergesucht. Der vorliegende Algorithmus ist nicht heuristisch. Es werden alle möglichen Belegungen gebildet und im nachhinein getestet, ob eine Lösung vorliegt. Der Suchalgorithmus kann durch den Einbau verschiedener heuristischer Algorithmen effektiviert werden. Zweige im Suchbaum, die ohnehin nicht mehr zu einer Lösung führen können, sollten möglichst zeitig abgeschnitten werden (z. B. können von vornherein Belegungen mit zwei unmittelbar nacheinander aufgestellten Damen auf der gleichen Waagerechten, Senkrechten oder Diagonalen ausgeschlossen werden). Den Einbau derartiger (problemabhängiger) heuristischer Algorithmen in einen vorhandenen Suchalgorithmus ermöglicht das Programm F00 „selbständig" mit Instruktion von einem Lehrer.

Lernen durch Analogie

Lernen durch Analogie ist ein Lernverfahren, das die menschliche Denkweise des Analogieschlusses beim Problemlösen auf maschinelle Lernsysteme überträgt. Mit solchen Lernsystemen kann ein vorhandenes Computerprogramm in ein anderes, das eine ähnliche, analoge Aufgabe erfüllt, umgewandelt werden. Zu diesem Zweck muß vorhandenes Wissen aus dem Speicher abgerufen, in geeigneter Weise transformiert, auf die neue, analoge Situation angewendet und für die künftige Benutzung wieder gespeichert werden.

Das Lernen durch Analogie wurde bereits sehr erfolgreich auf praxisrelevante Probleme angewendet, so zum Beispiel beim Finden neuer chemischer Verbindungen, die analoge Strukturen oder Eigenschaften zu bereits existierenden aufweisen. Aber auch für geometrische und zahlentheoretische Beweise kann dieses Lernverfahren benutzt werden, was anhand des folgenden elementaren Beispiels aus dem Mathematikunterricht illustriert werden soll /4/:

Zu Beginn wird einem Lernsystem (oder auch einem Schüler) der Beweis für folgende Behauptung vorgeführt:

Behauptung: Das Produkt zweier gerader Zahlen ist gerade.
Beweis:
1. Jede gerade Zahl ist durch 2 teilbar.
2. Die beiden geraden Zahlen seien x und y und können wie folgt dargestellt werden:
$x = 2m, y = 2n$ (m, n ganz)
3. Für das Produkt ergibt sich:
$x * y = 2m * 2n = 2(2mn)$ (2mn ganz)
4. Folglich ist das Produkt $x * y$ eine gerade Zahl.

Im Anschluß daran ist die folgende (analoge) Behauptung zu beweisen.

Behauptung: Das Produkt zweier ungerader Zahlen ist ungerade.
Beweis:
Der Beweis verläuft analog zum vorhergehenden, was sowohl von Schülern als auch vom zur Verfügung stehenden Lernsystem ausgenutzt wurde:
1. Ungerade Zahlen lassen bei Division durch 2 den Rest 1.
2. Die beiden ungeraden Zahlen seien x und y und können wie folgt dargestellt werden:
$x = 2m + 1, y = 2n + 1$ (m, n ganz)
3. Für das Produkt ergibt sich:
$x * y = (2m + 1) * (2n + 1)$
$= 4mn + 2m + 2n + 1$
$= 2(2mn + m + n) + 1$ $(2(mn + m + n)$ ganz)
4. Folglich ist das Produkt $x * y$ eine ungerade Zahl.

Die Analogie beider Beweise ist offensichtlich. Die Probleme derartiger Analogieschlüsse bestehen für den Computer im „Erkennen" der Analogie zwischen beiden Aufgabenstellungen und in ihrer Nutzung für den Problemlösungsprozeß.

Lernen aus Beispielen

Lernen aus Beispielen ist die bisher am intensivsten erforschte Methode des maschinellen Lernens. Vorgegeben ist zunächst eine Menge von Beispielen (auch positive Beispiele genannt) und/oder Gegenbeispielen (negative Beispiele) eines gewissen Begriffs (im allgemeinsten Sinne). Die Aufgabe des Lernenden besteht darin, allgemeinere Begriffsbeschreibungen zu erzeugen, die alle positiven Beispiele beschreiben und negative Beispiele nicht enthalten. Der Umfang an Inferenzen des Lernenden ist wesentlich größer als beim Lernen durch Instruieren und etwas größer als beim Lernen durch Analogie. Man unterscheidet prinzipiell zwei Arten des Lernens aus Beispielen:
– Alle Beispiele liegen bereits zu Beginn des Lernprozesses vor.
– Der Lernprozeß verläuft inkrementell, das heißt, die Beispiele werden nacheinander angeboten (auch induktives Lernen oder induktive Inferenz genannt).

Der zweite, weitaus häufigere Fall entspricht der induktiven Vorgehensweise des Menschen und soll im weiteren ausschließlich betrachtet werden. Der induktive Lernprozeß beginnt gewöhnlich mit einer leeren Beispielmenge. Die dazugehörige Regelmenge (zur Beschreibung allgemeinerer Begriffe) ist selbstverständlich auch leer. Neue Beispiele führen gegebenenfalls zu einer Erweiterung oder Änderung der bestehenden Regelmenge. Das induktive Lernen ist ein typischer Vertreter für einen Lernschritt nach Bild 2. Es soll anhand des folgenden Beispiels erläutert werden /5/:

Lernen von Regeln zur Pluralbildung englischer Substantive
Einem Lernsystem (oder Schüler) werden vom Benutzer (Lehrer) zunächst Beispiele vorgegeben, die immer aus den beiden folgenden Komponenten bestehen:
1. Komponente (Ausgangsform): Zeichenkette, die eine Basis für eine aus ihr abzuleitende Zeichenkette darstellt (z. B. Einzahl im Englischen: cat).
2. Komponente (Zielform): Zeichenkette, die aus der Ausgangsform abgeleitet werden kann (z. B. Mehrzahl im Englischen: cats).

Auf der Grundlage der angegebenen Beispiele soll das Lernsystem dann selbständig Regeln zur Pluralbildung aufstellen (1. Schritt in Bild 2) und anschließend aus den eingegebenen Ausgangsformen Zielformen erarbeiten (2. Schritt in Bild 2). Das notwendige Hintergrundwissen des Lernsystems beschränke sich auf einige „Kenntnisse" zur Verarbeitung von Zeichenketten (z. B. Vergleich von zwei Zeichen, Längenvergleich zweier Zeichenketten, positionsmäßiger Zeichenvergleich zweier Zeichenketten).

Wie können nun anhand vorgelegter Beispiele Regeln zur Pluralbildung hergeleitet werden? Wir kennzeichnen Mengen von Zeichenketten durch signifikante Teilzeichenketten, die in allen Elementen vorkommen (Notation: t2(...)). Unterschiedliche Teilzeichenketten innerhalb der Elemente einer Menge werden durch Variablen substituiert (Notation: t1(...)). Regeln arbeiten bei der Darstellung von Mengen von Zeichenketten nach folgendem Prinzip:
– Signifikante Teilzeichenketten des Definitionsbereiches werden auf signifikante Teilzeichenketten des Wertebereiches abgebildet.
– Nichtsignifikante Teilzeichenketten verbleiben unverändert.

Zur Erläuterung dient die Regel
r1 : t1(X),t2(y) – → t1(X),t2(ies)
Diese Regel besagt, daß in (gewissen) auf **y** endenden Zeichenketten das **y** durch **ies** ersetzt wird. r1(lady) führt damit zur Zerlegung in t1(lad),t2(y). Diese Zerlegung wird abgebildet auf t1(lad),t2(ies) und zu **ladies** zusammengesetzt, das heißt: r1(lady) = ladies.

„Was hast du gegen Künstliche Intelligenz? Mit deiner natürlichen bist du doch auch noch nicht weit gekommen, oder?"

Zeichnung: Frank Steger

Generalisierungsregeln (ein Prinzip des induktiven Lernens) erweitern die Definitions- und Wertebereiche von Regeln. Dies geschieht durch Verallgemeinerung (Generalisierung) der signifikanten Teilzeichenketten in den Mengen. Ziel ist es dabei, aus einer trivialen Anfangsregel, die genau die gegebenen Beispiele abbilden kann, eine allgemeinere Regel zu gewinnen. Als Beispiel dient Bild 3. Analog kann der Wertebereich dieser Regel
{ladies,bodies,ponies} zu t1(X),t2(ies) generalisiert werden. Damit wird durch Anwendung von Generalisierungsregeln aus der Regel
r: t2(lady); t2(body); t2(pony) → t2(ladies); t2(bodies); t2(ponies) die neue, allgemeinere Regel
r': t1(X),t2(y) → t1(X),t2(ies)

In der beschriebenen Art und Weise kann man Generalisierungsregeln auf vorgelegte Beispiele anwenden.

Mögliche Beispielpaare sind:

	Einzahl (Ausgangsform)	Mehrzahl (Zielform)
(1)	cat	cats
	dog	dogs
	day	days
	boy	boys
(2)	lady	ladies
	body	bodies
	pony	ponies
(3)	mouse	mice
	man	men

Folgende Regeln können daraus abgeleitet werden:
Man bildet den Plural englischer Substantive, indem man
(1) ein **s** ans Wortende anhängt
(2) ein **y** am Wortende in **ies** verwandelt, falls am Wortende ein Konsonant und **y** stehen
(3) aus **mouse mice** und aus **man men** bildet, da es sich hier um Ausnahmen handelt.

Ergebnisse erfolgreicher Generalisierungen liegen in den Regeln (1) und (2) vor. Ist keine Generalisierung möglich, so werden die Beispiele als Ausnahmen registriert (Regel (3)), das heißt, man verwendet die triviale Form einer Regel (Aufzählung).

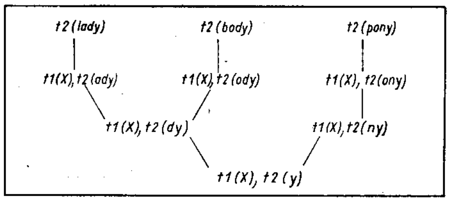


***Bild 3** Generalisierung des Definitionsbereiches* {**lady,body,pony**}

Lernen durch Beobachten und Experimentieren

Bei diesem Verfahren handelt es sich um die gegenwärtig allgemeinste Form des maschinellen Lernens (ohne die Hilfe eines Lehrers). Bei manchen Autoren (z. B. /6/) wird zwischen Lernen durch Beobachten und Lernen durch Experimentieren (eine noch höhere Form als Lernen durch Beobachten) unterschieden. Beide Methoden erfordern vom Lernenden mehr Inferenzen als in allen bisher behandelten Zugängen. Der Lernende erhält weder eine Beispielmenge noch spezielle Begriffe geliefert, sondern muß sich aus einem umfangreichen Datenmaterial selbständig typische Eigenschaften auswählen, neue Merkmale generieren, Theorien aufstellen, diese Theorien bestätigen oder widerlegen, und im Falle des Lernens durch Experimentieren sogar selbst Schritte zur Ausführung von Experimenten und zur Generierung zusätzlicher Beispiele mit dem Ziel der Wissensverbesserung einleiten. Diese höchste Form des maschinellen Lernens ist bisher relativ wenig erforscht.

Klassifikation auf der Basis der Wissensdarstellung

Die Darstellungsform in gegenwärtigen maschinellen Lernsystemen reicht von klassischen Methoden (Parameter in algebraischen Ausdrücken, Entscheidungsbäume, formale Grammatiken, Graphen, Netzwerke) bis hin zu modernen Methoden der Künstlichen Intelligenz (Produktionsregeln, Frames), was hier jedoch nicht näher ausgeführt werden soll.

Klassifikation entsprechend dem Anwendungsgebiet

Maschinelle Lernmethoden kamen bisher in den unterschiedlichsten Anwendungsgebieten zum Einsatz, an dieser Stelle seien nur einige genannt: medizinische und technische Diagnose, Chemie, Mathematik, Verarbeitung natürlicher Sprache, Spiele.

Die theoretische Erforschung maschineller Lernmethoden wurde auch durch Wissenschaftler der DDR maßgeblich beeinflußt, insbesondere seien hier die Arbeiten von Jantke, Thiele und Wiehagen von der Humboldt-Universität zu Berlin erwähnt (siehe z. B. /7/, /8/, /9/).

Literatur

/1/ Hörz, H. u. a. (Hrsg.): Wörterbuch Philosophie und Naturwissenschaften. Berlin: Dietz Verlag 1983, S. 513
/2/ Michalski, R. S.; Carbonell, J. G.; Mithchell, T. M.: Maschine Learning – An Artificial Intelligence Approach. Palo Alto, Cal.: Tioga Publishing Comp. 1983
/3/ Mostow, D. J.: Machine transformation of advice into a heuristic search procedure. In: /2/, S. 367
/4/ Carbonell, J. G.: Learning by analogy: formulating and generalizing plans form past experience. In: /2/, S. 137
/5/ Jansen-Winkeln, R. M.: Induktives Lernen von Grammatikregeln aus ausgewählten Beispielen. In: Savory, S. E. (Hrsg.): Künstliche Intelligenz und Expertensysteme. München, Wien: R. Oldenbourg Verlag 1985, S. 211
/6/ Fohmann, L.: Wissenserwerb und maschinelles Lernen. In: Savory, S. E. (Hrsg.): Künstliche Intelligenz und Expertensysteme. München, Wien: R. Oldenbourg Verlag 1985, S. 125
/7/ Jantke, K. P.: Ansätze für die Software-Entwicklung aus unvollständigen Spezifikationen. 4. Symposium Grundlagen und Anwendung der Informatik, Technische Hochschule Karl-Marx-Stadt, 4.–6. Febr. 1986
/8/ Jantke, K. P.: The main proof theoretic problems in inductive inference. In: Frege Conference 1984, Proc., G. Wechsung (Ed.), Berlin: Akademie-Verlag 1984, S. 321
/9/ Klette, R.; Wiehagen, R.: Resarch in the theory of inductive inference by GDR mathematicians – a survey. Information Sciences 22 (1980), S. 149

✉ KONTAKT ☎

Technische Universität Karl-Marx-Stadt, Sektion Informatik, PSF 964, Karl-Marx-Stadt, 9010; Tel. 66 82 52

Systeme mit System — Teil 2

Vorausschauen vor dem Erweitern

Detlef Grell

„Ist eine gewünschte Erweiterung überhaupt möglich, und wenn ja, wie?" lauten zwei der wichtigsten Fragen vor einem PC-Kauf. Um dem Händler konkrete Fragen vorlegen zu können, stellt man sich zum einen das zukünftige „Vollsystem" tabellarisch zusammen, zum andern die Version, die bei der Anschaffung derzeit ausreicht.

Leider kennt ein Händler auch nicht immer alle Tücken seines Rechnerangebots, so daß die Kenntnis der wichtigsten Erweiterungs-Hemmnisse eine gute Argumentationsgrundlage bildet. Sollten nach gündlicher Aussprache mit dem Verkäufer bei Ihnen Zweifel verblieben sein, so lassen Sie sich die wichtigsten Optionen schriftlich garantieren.

XT-Fallen

Die typischen Probleme mit XT-Motherboards wurden bereits aufgeführt:
Integrierte Floppy- oder Harddisk-Controller und Grafikadapter, die sich nicht ausblenden lassen, verhindern den Einsatz verbesserter Varianten unter Umständen nachhaltig. Es gibt auch Floppy- oder Harddisk-Controller, die man auf verschiedene Adressen einstellen kann – die aber von den meisten BIOS-Versionen ignoriert werden. In solch fragwürdigen Fällen daher immer ein Rückgaberecht für die Neuerwerbung vereinbaren.
Bei Erweiterungen mit seriellen oder parallelen Schnittstellen-Karten gibt es selten Probleme, wenn man darauf achtet, daß sich die neuen Karten auf verschiedene Portadressen legen lassen.

Offen, aber kompliziert

Das Hauptproblem bei ATs aller Schichten, also einschließlich 80386, ist die RAM-Be-

Fertiglösung oder Einzelteile?

Es ist zwar reizvoll, aber recht problematisch, sich einen Rechner vollständig aus Wunsch-Komponenten selbst zusammenzukaufen, um ihn dann Teilchen für Teilchen in Betrieb zu nehmen. Gerade bei AT-Rechnern ist das Bustiming nicht exakt von IBM spezifiziert, so daß es zwischen den Nachahmern relativ große Abweichungen gibt. Damit ist grundsätzlich nicht gewährleistet, daß jede Steckkarte in jedem Rechner läuft. Zum ernsthaften Problem wird das immer dann, wenn die Grafikkarte nicht mitspielt, weil dann die Inbetriebnahme aller weiteren Komponenten „wegen Blindflug" ins Stocken gerät.

Dabei tun sich nicht zwangsweise Motherboards mit hohen CPU-Taktraten besonders negativ hervor, sondern meist sind es Boards, deren Bustakt-Frequenz nicht auf 8 oder 10 MHz absenkbar ist. Im Grunde muß man bei jeder Steckkarte, bei der der Hersteller nur das Zusammenspiel mit einem IBM AT (8 MHz, 1 Wait-State) garantiert, mit Problemen rechnen, wenn man sie auf deutlich schnelleren Rechnern einsetzt.

Das gilt noch viel mehr, wenn man gedenkt, günstig erstandene Karten zu verwenden, die bislang nur in einfachen XTs erprobt wurden. Zwar stellt heute niemand mehr zwei Kategorien Steckkarten mit unterschiedlichen Geschwindigkeiten für ATs und XTs her, aber die Gefahr ist sehr groß, daß ältere Steckkarten, die nur für XTs gefertigt wurden in schnellen ATs versagen.

Es ist jedem daher nur zu empfehlen, sich seine Wunschkombination vom Händler konfigurien zu lassen, der dann sehen muß, wie er ein lauffähiges System zustandebringt. Schlimmstenfalls wird er eingestehen, daß bestimmte Steckkarten für den Zielrechner nicht geeignet sind, und wird Alternativen empfehlen.

Kann der „Einzelteilkäufer" einen sachkundigen Berater hinzuziehen und steht noch dazu ein vergleichbarer, lauffähiger Rechner als Testreferenz zur Verfügung, dann kann man schon eher einen „Eigenbau" wagen. Man hat es in der Regel aber nicht leicht, ungeeignete Komponenten an Händler zurückzugeben, wenn man solche Optionen nicht vorher schriftlich vereinbart hat.

stückung. Die meisten 80286-ATs lassen nicht mehr als 1 MByte RAM auf dem Motherboard zu, aufgeteilt in 640 KByte für DOS und 384 KByte Extended Memory (also jenseits von 1 MByte, wo keine DOS-Programme laufen).
Bei einigen älteren ATs (auch Commodores erstem PC 40) waren allerdings nur die Varianten 512 KByte für DOS und 512 KByte Extended RAM oder 640 KByte für DOS – und 0 KByte für Extended Memory möglich. Die Lösung für 640 KByte DOS-RAM ohne Verluste besteht darin, die 512-512-Aufteilung zu benutzen und die fehlenden 128 KByte DOS-RAM mit einer zusätzlich zu kaufenden RAM-Karte aufzufüllen.
8-Bit-RAM-Karten im 16-Bit-AT bremsen die CPU durch geringe Busbreite und etliche Wait-States, wenn 16-Bit-Zugriffe in aufeinanderfolgende 8-Bit-Zugriffe umzuwandeln sind. 16-Bit-Karten sind auch deshalb die richtige Wahl, weil nur die „langen Slots" die Adreßleitungen für den Extended-Memory-Bereich führen.
Die meisten ATs mit 0 Wait-States bedenken übrigens auch RAM-Karten in 16-Bit-Slots mit einem Wait-State. In neueren Rechnern, vornehmlich bei den 386ern, gibt es daher ausschließlich für RAM-Erweiterungen mit 0 Wait reservierte Spezial-Steckplätze.

Extended oder Expanded?

Ein AT mit 80286 oder 80386SX vermag 16 MByte RAM zu adressieren, etliche Motherboards fassen (nebst Zusatzkarten) aber nur maximal 8 MByte. Das ist zwar nicht direkt bedenklich, aber schon heute handeln „fortschrittliche Power-User" 8 MByte als gerade richtig für die OS/2-Vollversion und Unix.
Die Mehrheit der Anwender hat jedoch schon Schwierigkeiten, 4 MByte RAM speziell auf 80286-ATs unter DOS sinnvoll einzusetzen. Denn Extended Memory können die wenigsten Programme direkt nutzen (Stichwort „DOS-Extender"), meist dient er nur als riesige RAM-Disk. Interessanter für einen 80286-AT ist hingegen eine RAM-Erweiterung, die wahlweise als Extended oder Expanded Memory nutzbar ist (sogenannte Above Boards).
Expanded Memory ist hardwaremäßig etwas grundsätzlich anderes als Extended Memory, denn es stellt zusätzliches RAM im Adreßraum unterhalb 1 MByte dar, ist also auch XTs zugänglich. Dabei können etliche MByte RAM „portionsweise" in 16-KByte-Blöcken in den Adreßbereich zwischen Bildspeicher und ROM-BIOS eingeblendet werden. Man hat jedoch nur Vorteile vom Expanded Memory (genau wie bei Koprozessoren), wenn die benutzte Anwender-Software damit umgehen kann. Allerdings gibt es erheblich mehr DOS-Software, die Expanded Memory nutzt als Extended.
ATs mit NEAT-Chip-Satz können ihren Speicher wahlweise als Extended oder Expanded Memory konfigurieren. Auf 80386-Rechnern kann aufgrund der CPU-Fähigkeiten Extended Memory per Software (80386-EMS-Treiber) auch als Expanded Memory eingerichtet werden.

Megabit-Falle

Das Motherboard-Design prägt das Puzzle-Spiel, wie man heute wenig und morgen viel RAM auf den Rechner bringt, ohne alte Chips ausmustern zu müssen. Wenn die Hauptplatine zum Beispiel nur auf 2 MByte aufrüstbar ist und darüber eine Zusatzkarte nötig wird, dann sollte man die Zusatzkarte unbedingt gleich mitkaufen. Denn diese ist vielfach „nach Art des Hauses" mechanisch und elektrisch auf ein ganz bestimmtes Motherboard zugeschnitten und vielleicht übermorgen schon nicht mehr lieferbar.
Des weiteren ist zu klären, ob die Motherboard-RAMs zusätzlich zu denen auf der Erweiterungskarte betrieben werden können oder nur statt dessen. In ganz unglücklichen (aber real existierenden) Fällen muß man die Board-RAMs nicht nur aus der Hauptplatine entfernen, sondern diese passen nicht mal auf die Zusatzkarte.
Typische 80386DX-Motherboards lassen sich wahlweise mit Megabit-Chips auf 8 MByte und mit 256-KBit-RAMs auf 2 MByte bestücken. Wenn pro Bit ein RAM-Chip (× 1-Organisation, es gibt auch × 4) fällig wird, dann muß man mit Megabit-Chips minimal 4 MByte bestücken, damit man die Busbreite der CPU erreicht und der Rechner überhaupt laufen kann. Begnügt man sich bescheiden mit 2 MByte via 256er Chips, sind die RAM-Fassungen auf dem Motherboard voll: Eine Aufrüstung nur auf der Hauptplatine kann jetzt lediglich durch Ersetzen eines MBytes 256er durch Megabit-Chips erfolgen – die 256er hat man danach übrig. Wenn eine Wahlmöglichkeit bei der RAM-Geschwindigkeit besteht (0 oder 1 Wait-State), so zahlen Sie lieber den relativ geringen Aufpreis für die schnelleren RAMs. Später müßten Sie zur Erlangung eines 25 Prozent schnelleren Rechners alle vorhandenen RAMs ersetzen.
Einige Rechner-Boards lassen auch einen Betrieb mit „fast 0 Wait-State" durch sogenanntes Bank-Interleaving zu. Dabei muß lediglich gewährleistet sein, daß zwei vollständige RAM-Blöcke bestückt sind. Eine Bank ist dabei die Anzahl RAMs, die minimal nötig ist, damit der Rechner läuft. Im obigen Beispiel 80386DX mit Megabit-Chips (× 1) stellen 4 MByte erst eine Bank dar, bei einem 80386SX oder 80286 halbiert sich dieser Wert durch die Busbreite. Für Hobbyisten ist auch daher ein 80386SX eine attraktive Wahl.
Es lohnt sich also, beim Rechnerkauf genau zu klären, wie es später weitergeht.

Äußerlich und innerlich

Auch das Gehäuse setzt die Grenzen der Erweiterbarkeit für Laufwerke. Die Controller in PCs und ATs können standardmäßig zwei Floppy- und zwei Harddisk-Laufwerke ansteuern, nur Spezial-Controller erlauben mehr. Die kompaktesten Gehäuse haben entweder eine sehr geringe Grundfläche (Baby-Gehäuse) oder sind sehr flach (slim line). Oft fassen diese Gehäuse nur maximal drei Laufwerke halber Bauhöhe, so daß oberhalb der Standardausrüstung „eine Harddisk, zwei Floppies" nichts mehr geht.
Die meisten Gehäuse haben jedoch vier Einbauplätze für die heute üblichen Laufwerke mit halber Bauhöhe, wenn auch mit sehr unterschiedlichem Komfort, was das schnelle Wechseln der Laufwerke betrifft: Je kompakter das Gehäuse, desto mühsamer.
Bei hoher Festplattenkapazität benötigt man Backup-Geräte (Band-Streamer, optische Platten). Diese haben meist eigene Controller, so daß sich sehr wohl mal ein fünftes Laufwerk ins Gehäuse verirren könnte. Dafür bieten einige, jedoch längst nicht alle Tower-Gehäuse Platz.
Tower-Gehäuse lassen sich angenehmerweise etwas weiter weg vom Ohr neben den Schreibtisch stellen. Allerdings liegt man bei jedem Platinenwechsel oder Abschrauben eines Steckers auf dem Fußboden, denn meist verhindert die Verkabelung, das Gerät mal eben auf den Tisch zu stellen. „Fummlern" ist daher ein Tischgehäuse mit Klappmechanismus zu empfehlen, jedoch gibt es gerade bei dieser Gehäuse-Bauform viele Produkte, die eher den Namen „Klappermechanismus" verdienen. Achten Sie auf solide Ausführungen, denn ein Gehäusedeckel, der mit Lüfter oder Festplatte „mitbrummt", nervt ungemein.

Aug in Aug

Grafik von CGA bis Super-VGA

Detlef Grell, Manfred Bertuch

Am kompliziertesten geht es derzeit auf dem Sektor der Video-Karten für PCs zu. Man unterscheidet sie anhand ihrer höchst unterschiedlichen Auflösungen und Ablenkfrequenzen – wofür Sie anschauliche Beispiele auf der 2. Umschlagseite finden –, beides wichtige technische Daten, die auch den anzuschließenden Monitor spezifizieren. Der Monitor selbst ist eine der wichtigsten Schnittstellen des Computers zum Menschen. Nichts macht ein Rechnersystem so unbrauchbar wie ein Monitor, der sich für die geplanten Anwendungen als ungeeignet erweist.

CGA

Der Color Graphics Adapter (CGA) ist der älteste PC-Standard mit der geringsten maximalen Auflösung von 640 × 200 Bildpunkten. Seine Ablenkfrequenzen (15750 Hz horizontal, 60 Hz vertikal) wurden vermutlich so niedrig gewählt, um auch Fernsehgeräte als Monitor verwenden zu können, üblich sind jedoch RGB-Farbmonitore und Monochrom-BAS-Monitore. Die aus heutiger Sicht in jeder Betriebsart unzureichende Bildqualität verbietet den CGA-Einsatz nicht nur in professionellen Systemen.

MDA und Hercules (MGA)

IBMs Monochrom Display Adapter MDA ermöglicht ausschließlich Textdarstellung mit einer Auflösung von 720 × 350 Bildpunkten. Die Schriftzeichen lassen sich relativ fein (9 × 14 Punkte) auflösen, und es verbleibt zwischen den Zeichen – ganz anders als bei CGA – genügend Abstand, um Texte ohne Mühe lesen zu können. Der Monochrom Graphics Adapter (MGA) der Firma Hercules bietet zusätzlich zum MDA-Textmodus auch Grafik mit 720 × 350 Bildpunkten und hat den MDA heute quasi vom Markt verdrängt.
Eine Kombination aus MGA-Karte und TTL-Monochrom-Monitor ist preiswert (zusammen rund 300 DM), recht schnell, und ihre Qualität reicht für die meisten professionellen Anwendungen aus. Vor allem das kontrastreiche und scharfe Bild eines guten TTL-Monitors ist den meisten Farbmonitoren um einiges überlegen. Lediglich vom „inversen Betrieb", also dunkle Schrift auf hellem Hintergrund, muß man abraten: Diese Variante flimmert wegen der 50-Hz-Bildfrequenz (18430 Hz horizontal) unerträglich.

AGA

Advanced Graphics Adapter hieß eine Video-Karte von Commodore, die bald Nachahmer fand (etwa Everex, auch im Schneider Euro PC). AGA ist eine technische Synthese aus CGA und Hercules-Karte, die in der jeweiligen Betriebsart auch *nur* CGA- oder *nur* TTL-Monitore korrekt ansteuert. Beachten Sie: Einfache CGA- oder TTL-Monitore darf man nicht mit Zeilenfrequenzen oberhalb ihrer Spezifikation betreiben, weil sie dadurch (Anstieg der Hochspannung) zerstört werden können. Inzwischen gibt es aber auch Zwei-Norm-Monitore, die sich automatisch (und gefahrlos) auf die beiden Betriebsarten eines AGA einstellen.

EGA

Der Enhanced Graphics Adapter ist heute die preiswerteste, jedoch schon etwas veraltete Lösung für augenverträgliche PC-Farbgrafik. Mit 640 × 350 Bildpunkten erreicht er fast MDA-Qualität, 16 aus 64 Farben gleichzeitig werden geboten. Mit der EGA-Karte brachte IBM einen speziellen Zweifrequenz-Monitor heraus, der CGA- und EGA-Auflösung beherrscht. Bei letzterer beträgt die Zeilenfrequenz 21850 Hz, die Bildwiederholfrequenz 60 Hz.
Eine EGA-Karte kann mit einem CGA-Farbmonitor zusammenarbeiten und steuert diesen dann mit den richtigen Signalen an, dies aber natürlich auch nur in CGA-Auflösung (640 × 200). Der Original-IBM-EGA ist BIOS-, aber nicht völlig registerkompatibel zu CGA. Weiterhin unterstützt EGA auch den echten MDA-Modus mit zugehörigen Ablenkfrequenzen, erfordert dann also einen TTL-Monochrom-Monitor.
Einen guten EGA-Monitor mit so gestochen scharfem Bild und entspiegelter Bildröhre wie das Gerät von Sakata, das wir Anfang 1987 bei einem Test loben konnten, muß man heute lange suchen: Es gibt fast nur noch Low-End-Produkte. Wird man dabei nicht fündig und möchte seinen Augen zuliebe einen Multinorm-Monitor anschaffen, dann ist eine Lösung aus VGA-Karte und VGA-Monitor preislich etwa gleichwertig, technisch aber sicher die bessere und zukunftsträchtigere Wahl.
EGA-Karten vieler Fremdhersteller können zusätzlich auch MGA emulieren (an TTL-Monitoren) und sind völlig CGA-kompatibel (für Spiele). Die augenfälligsten Verbesserungen bestehen in weitaus höheren Auflösungen (bis 800 × 600, Super-EGA). Diese Modi sind aber nicht von IBM standardisiert und daher bei jedem Hersteller anders ausgeführt. Man kann sie nur in Verbindung mit Programmen anwenden, für die es zugehörige Treiber gibt. Außerdem braucht man aufgrund höherer Ablenkfrequenzen in diesen Modi auch einen Multinorm-Monitor. Dieser kostet zwar etwas mehr als ein purer EGA-Monitor, ist aber auch deshalb die bessere Wahl, weil man ihn bei einem Umstieg auf VGA weiterverwenden kann.

MCGA

Das Multi Color Graphics Array ist eine abstruse Halb-VGA auf dem Motherboard des IBM Modell 30. Sollte es MCGA-Steckkarten geben, so ignorieren Sie diese zugunsten vollwertiger VGA-Karten. VGA, nicht MCGA heißt der akzeptierte neue Standard.

VGA

Das Video Graphics Array implementierte IBM auf dem Motherboard seiner PC/2-Rechner ab dem Modell 50. VGA hat – anders als alle seine digitalen Vorgänger – ein Analog-Interface bekommen, das Farbschattierungen beziehungsweise auf Schwarzweißmonitoren Graustufen erlaubt. VGA erfordert Monitore mit Analog-Eingang für deutlich erhöhte Ablenkfrequenzen (horizontal 31500 Hz), wobei die 70 Hz Bildablenkfrequenz die Darstellung von schwarzer Schrift auf weißem Hintergrund zuläßt.
Die höchste Auflösung beträgt bei 16 aus 262144 möglichen Farben im Textmodus 720 × 400, im Grafikmodus 640 × 480 Bildpunkte. In einem Grafikmodus mit 320 × 200 Bildpunkten können gleichzeitig 256 Farben (64 Graustufen) aus 262144 dargestellt werden. CGA-, MDA- und EGA-Kompatibilität ist nur noch auf Software-Ebene gegeben, wodurch aber deren Darstellung auf dem VGA-(Festfrequenz-)Monitor möglich ist. Üblich sind VGA-Monitore mit 12 bis 14 Zoll Schirmdiagonale, die mit den von IBM gesetzten Maximalauflösungen ein zufriedenstellendes Bild liefern. Zu diesem Monitor-Typ gehört übrigens auch ein Gerät mit dem irreführenden Namen „MultiSync 2A" von NEC.

Super-VGA

Ein Teil der VGA-Nachbauer stellt auch Karten mit voller Registerkompatibilität zu älteren Adaptern her, manche Karten haben zusätzlich zum neuen 15poligen VGA-Stecker ebenfalls den alten TTL-Stecker zum Anschluß der Vorgänger-Monitore. Auch findet man heute AT-gerechte VGA-Karten mit 16-Bit-Bus.
Super-VGAs lösen bis zu 800 × 600 Punkte bei 256 Farben und 1024 × 768 mit 16 Farben auf. Diese Modi eignen sich durchaus schon für einfache CAD-Anwendungen oder „Privat-DTP". Das erfordert allerdings einen Bildspeicher von 512 KByte statt der bei IBMs VGA üblichen 256 KByte und ebenso den Anschluß eines Multinorm-Monitors, weil dabei deutlich höhere Ablenkfrequenzen (35 bis 48,5 kHz) anfallen.
Unserer Meinung nach ist der 1024 × 768-Modus mit Zeilensprungverfahren (flimmert) und an Monitoren mit einer Schirmdiagonale unter 16 Zoll nicht attraktiv (Linien und Schrift erscheinen für das Auge beim üblichen Betrachtungsabstand einfach zu klein). Eine vernünftige Lösung ohne Zeilensprungverfahren (non-interlaced) mit einem 19-Zoll-Monitor ist aber für gelegentliche Farbspielereien zu teuer. Man muß daher vor dieser Investition genau prüfen, ob man diese Auflösung tätsächlich häufig braucht und ob für die anvisierte Anwendung überhaupt ein Treiber für die Auflösung erhältlich ist.
Der 800 × 600-Modus ist – allemal für Hobbyisten – viel attraktiver, weil der sich auch auf handelsüblichen Multinorm-Monitoren (knapp über 1000 DM) recht gut macht. Es existieren dabei zwei grundverschiedene Modi, wobei der eine *nicht* eine abgemagerte Variante des andern ist. 800 × 600 bei 16 Farben bieten auch schon einfache 8-Bit-Karten mit 256 KByte Video-RAM, 800 × 600 mit 256 Farben erfordern 512 KByte RAM. Ziehen Sie jetzt aber nicht den verhängnisvollen Rückschluß, daß eine VGA-Karte mit 512 KByte automatisch die beiden genannten Modi unterstützt: Solche gibt es natürlich (etwa Eizo MD-B10), aber vor allem bei Karten, die auch 1024 × 768 auflösen, sparen die Anbieter gern einen der 800 × 600-Modi ein.
14-Zoll-Multiscans bieten sich auch immer dann an, wenn man einen Monitor zeitweise an unterschiedlichen Karten (CGA, Hercules, EGA) oder sogar an verschiedenen Rechnern (Archimedes, Atari ST oder Amiga) betreiben will. Multinorm-Monitore sind aber nicht alle gleich. Die kleineren und preiswerteren 14-Zoll-Geräte (bis 800 × 600) synchronisieren bis hinab in den CGA- und TV-Bereich. Einige kostspieligere Versionen

Fortsetzung auf Seite 115

Einführung in MS-DOS

Teil 1

Wolfram Schulze, Uwe Schulze, Berlin

Viele MS-DOS-Einsteiger sind durch die vorhandene Literatur, die zumeist Wert auf Systematik und Vollständigkeit legt, nicht ausreichend bedient. Mit diesem Kurs soll deshalb der Versuch unternommen werden, Ihnen eine didaktische Arbeitshilfe zu bieten, in die die Erfahrungen der Arbeit mit dem Betriebssystem einfließen. Deshalb sind die Befehle auch nicht – wie oft üblich – alphabetisch geordnet, sondern nach praktischen Gesichtspunkten. Sie werden viele Beispiele finden, jedoch auch wertende, warnende, vor allem aber ermunternde Hinweise. Vorkenntnisse sind nicht nötig, ein paar Erfahrungen von CP/M oder im Umgang mit Standardsoftware würden das Verständnis der oft knappen Darstellung aber erleichtern.

Die ersten Schritte

Der erste Teil zeigt, wie das System gestartet wird und beschreibt das nicht ganz einfache DOS-Dateisystem. Die ersten Kommandos beziehen sich auf die Arbeit mit der Diskette. Schwerpunkt: Arbeit mit Unterverzeichnissen.

Historie

Als der größte Computerhersteller der Welt, die International Business Machines Corporation (IBM) 1980 nach einem Entwickler für das Betriebssystem (die Grundsoftware) ihres ersten 16-Bit-Personalcomputers suchte, kam dafür eigentlich nur Digital Research mit seinem Produkt CP/M-86, dem Nachfolger des erfolgreichsten kommerziellen Betriebssystems für 8-Bit-Technik, in Frage. Doch das Rennen machten Microsoft und sein MS-DOS (Microsoft Disc Operating System). Hierbei handelt es sich um ein von der Firma Seattle Computer Products unter dem Namen QDOS (Quick and Dirty Operating System) entwickeltes System mit Anteilen bei Microsoft (Arbeitsname DOS-86). Mit Markteinführung des IBM-PCs im August 1981 erwarb Microsoft alle Rechte, und gleichzeitig erschien auch PC-DOS, die identische IBM-Version von MS-DOS. Auch heute noch existieren beide Versionen nebeneinander; alle Nicht-IBM-PCs verwenden in der Regel MS-DOS.

Während seiner fast 10jährigen Geschichte unterlag MS-DOS einer ständigen Entwicklung, die besonders durch die Fortschritte bei der Hardware (z.B. Festplattenlaufwerke größerer Kapazität) hervorgerufen wurde. Leider scheint der ehemalige Name (Quick and Dirty = schnell und unsauber programmiert) dafür oftmals Programm gewesen zu sein.

Die erste DOS-Version (1.x) war CP/M noch sehr ähnlich. Mit der Version 2.x nahm es die Gestalt an, die uns auch heute noch vertraut ist. Dafür wurde eine Reihe erfolgreicher Prinzipien von Unix (einem in den Bell Telephone Laboratories, New Jersey, entwickelten portablen Betriebssystem mit modernen Konzepten) übernommen. Zu nennen wären das hierarchische Dateisystem, das in diesem Teil des Kurses noch zu besprechen sein wird), die Ein-/Ausgabe-Umleitung und die UNIX-ähnlichen Systemrufe, die heute fast ausschließlich verwendet werden. Viele der neueren Programme kommen auch noch mit dieser Version zurecht. Die Versionen 3.x brachten Funktionen für den Netzbetrieb (die Nutzung aus Turbo-Pascal heraus ist in MP 6/89 beschrieben), die Beachtung landesspezifischer Besonderheiten sowie die Unterstützung größerer Festplatten. Außerdem wurden einige vorhandene Systemkommandos erweitert.

Den gegenwärtigen Standard bilden die Versionen 3.2 und 3.3. Auf diese soll in diesem Beitrag auch Bezug genommen werden. Die inzwischen eingeführte Version 4.0 setzt sich nur zögerlich durch. Sie bringt uns neben der Nutzung großer Festplatten „in einem Stück" vor allem eine grafische Nutzeroberfläche. Statt der Eingabe von Kommandos werden Menüpunkte angeboten, die mit der Maus ausgewählt („angeklickt") werden.

Damit dürfte die Entwicklung von MS-DOS abgeschlossen sein. Grund dafür ist, daß aus Kompatibilitätsgründen – die alten CP/M-ähnlichen Funktionen werden immer noch unterstützt – die neuen Hardwaremöglichkeiten nicht genutzt werden können. Das bezieht sich besonders auf den Hauptspeicher oberhalb der 1-Mega-byte-Grenze und die neuen Möglichkeiten der Prozessoren (protected mode). Das Nachfolgesystem OS/2 beginnt sich langsam zu etablieren. Trotzdem wird DOS vorerst noch das am weitesten verbreitete Betriebssystem bleiben. Man spricht von 45 Millionen Installationen weltweit. Neben dem eigentlichen Betriebssystem hat sich noch eine Reihe von Erweiterungen entwikkelt, auf die in diesem Beitrag nicht eingegangen werden kann. Dazu gehören Nutzeroberflächen (wie MS-Windows und GEM), Netzschnittstellen und Möglichkeiten zur Nutzung eines größeren Speichers (wie Expanded Memory Specification). Im folgenden wird MS-DOS auch kurz als DOS bezeichnet.

Hardware

Herzstück des PCs ist ein Prozessor der Firma Intel. Beim IBM PC war es ein i8088, bei Kompatiblen meist ein i8086. Der einzige Unterschied besteht darin, daß der 8088 trotz interner 16-Bit-Architektur nach außen einen 8-Bit-Datenbus besitzt, so daß anfangs die alten Peripherieschaltkreise genutzt werden konnten. Sein Nachfolgemodell nannte IBM PC/XT (XT für eXtended Technology), dessen wichtigste Neuerung eine Festplatte war. Er – der Standard-PC unserer Tage – wurde inzwischen vom Modell PC/AT (AT für Advanced Technology) abgelöst. Er nutzt den Intel-Prozessor 80286 und im allgemeinen eine größere Festplatte. Damit sind vor allem Geschwindigkeitsvorteile verbunden. International vollzieht sich inzwischen auf breiter Front eine Ablösung durch den 80386, den ersten 32-Bit-Prozessor von Intel. Seine Stärken kann MS-DOS – wie schon erwähnt – nur unzureichend ausnutzen, deshalb bleibt er vor allem OS/2, UNIX und anderen leistungsfähigen Betriebssystemen vorbehalten.

Aber zurück zum Standard-PC. Vorwiegend ist er mit 640 KByte Hauptspeicher ausgerüstet. Das ist die obere Grenze für DOS. Viele Programme kommen auch mit wesentlich weniger Speicher aus, da die PCs der Vergangenheit zuerst mit 256, später mit 512 KByte ausgestattet waren. Doch der exzessive Umgang mit Speicherplatz hat inzwischen dazu geführt, daß für eine Reihe von Programmen die 640-KByte-Grenze schon Probleme mit sich bringt.

Start des Betriebssystems

Nach der Einführung wenden wir uns den ersten praktischen Schritten zu. Besitzt der PC eine Festplatte (und soviel sollten Sie über Ihren Computer schon wissen), so genügt zum Laden des Betriebssystems – der Fachmann sagt booten – das Einschalten. Wenn nicht, so wird der Anwender mit einer Ausschrift auf dem Bildschirm zum Einlegen der sogenannten Systemdiskette aufgefordert. Diese wird vom Hersteller mitgeliefert und ist als solche gekennzeichnet.

Laden Sie das System stets von der Diskette, so sollten Sie sich folgende Arbeitsschritte angewöhnen

- Einschalten
- Systemdiskette einlegen
- Reset-Taste betätigen.

Damit vermeiden Sie die eventuelle Zerstörung von Daten auf der Systemdiskette.

Anschließend fordert das Betriebssystem die Eingabe des aktuellen Tagesdatums und der Zeit durch folgende Ausschriften:

Beispiel:

```
Aktuelles Datum ist Die 11.07.1989
Neues Datum(mm.tt.jj):_
```

und

```
Uhrzeit: 12:06:45.18
Neue Uhrzeit:_
```

Vorausgesetzt, Sie arbeiten mit einer Betriebssystem-Version, die deutsche Ausschriften verwendet, ansonsten ergibt sich ein geringfügig anderes Bild. Das Datum ist wichtig für einige Systemfunktionen und sollte deshalb nicht durch Drücken der Enter-Taste übergangen werden. Hat Ihr Rechner keine batteriegepufferte Uhr, so wird er beim Einschalten den 01.01.1980, 00.00 Uhr – die Geburtsstunde des IBM PCs – anzeigen. Werden Sie nicht nach Datum und Zeit gefragt, so kann man dem abhelfen. Wie, das zeigt die nächste Folge.

Als nächstes erscheint die Copyrightausschrift des Betriebssystems und das Eingabebereitschaftszeichen (Laufwerksbuchstabe und Größerzeichen), Prompt genannt. Danach erscheint ein Unterstrich, der Cursor genannt wird. An dieser Position erscheint das nächste Zeichen, das über die Tastatur eingeben wird.

Beispiel:
Microsoft(R) MS-DOS(R) Version 3.30
(C)Copyright Microsoft Corp 1981–1987

A>_
Der Buchstabe symbolisiert das aktuelle Laufwerk, mit dem jetzt gearbeitet werden kann. Wurde das System von der Festplatte geladen, so erscheint der Buchstabe C (als logisch erstes Laufwerk der Festplatte). Nach dem Laden des Systems von der Diskette – wie in unserem Beispiel – wird das A angezeigt. In der Regel gilt folgende Zuordnung der Buchstaben zu den Laufwerken:
A erstes Diskettenlaufwerk
B zweites Diskettenlaufwerk
C Festplatte
D weiter logische Laufwerke
Es hängt von der vorhandenen Hardware ab (Größe der Festplatte, Anzahl der Diskettenlaufwerke), welche logischen Laufwerke Ihnen tatsächlich zur Verfügung stehen.
Einem physischen (wirklich vorhandenen) Laufwerk können mehrere logische (scheinbare) Laufwerke zugeordnet sein. So kann zum Beispiel das Laufwerk B: im DOS-Format und das gleiche Laufwerk als D: für ein CP/M-Format (800 KByte) genutzt werden (der Doppelpunkt hinter den Laufwerksbuchstaben kennzeichnet ihn als Gerät – im Gegensatz zu Dateien, die auch A oder B heißen könnten). Auch eine RAM-Disk (Simulation eines Laufwerkes im Hauptspeicher – oft als E: genutzt) ist möglich.
Das Umschalten auf ein anderes Laufwerk erfolgt mit dem Laufwerksbuchstaben und einem Doppelpunkt, sowie dem Betätigen der ENTER-Taste (im folgenden als <ENTER> dargestellt).
Zum Laufwerk C: geht es also mit der Eingabe von
c: <ENTER>
und zurück nach A: mit:
a: <ENTER>
Es ist gleichgültig, ob Sie große oder kleine Buchstaben verwenden. Das gilt für alle Kommandos im Betriebssystem. DOS wandelt alle Eingaben in große Buchstaben um. Auch die Dateinamen werden stets als Großbuchstaben angezeigt. Im folgenden werden hervorgehobene Kommandos mit Kleinbuchstaben fettgedruckt dargestellt.

Die Tastatur

Das Tastaturfeld ist nicht bei allen MS-DOS-fähigen Computern identisch, aber es gibt weitgehende Ähnlichkeit. Prinzipiell unterteilt sich das Tastenfeld in vier Bereiche: Buchstabentasten, Numerische Tastatur (separater Ziffernblock), Funktionstasten und Steuertasten.
Wichtige Funktionen haben die folgenden Tasten (das Einschließen in spitze Klammern kennzeichnet die jeweilige Taste):
<ENTER> Abschluß eines Kommandos
<SHIFT> Umschalten Groß-Kleinschreibung
<ESC> Löschen der Eingabezeile; in vielen Programmen Beenden des aktuellen Menüs
<ALT> Dient zur Eingabe erweiterter Codes.
Drücken Sie doch einmal die ALT-Taste und dann auf dem Ziffernblock rechts 42 – beim Loslassen der ALT-Taste erscheint ein *. Diese Möglichkeit wird noch einmal sehr wertvoll für Zeichen sein, die auf der Tastatur vermißt werden.
<CTRL> Ist gemeinsam mit anderen Tasten zu betätigen, z. B.
<CTRL> <H> ein Zeichen zurück, das gleiche wird auch mit dem Pfeil ← erreicht
<CTRL> <G> Löschen des aktuellen Zeichens (des Zeichens, auf dem der Kursor gerade steht)
<CTRL> <P> Zuschalten des Druckers, er druckt alles mit
<CTRL> <S> Anhalten des rollenden Bildschirms, dazu kommen wir noch
<PrtScr> druckt den aktuellen Bildschirm aus (Hardcopy)
Die Bedeutung der Funktionstasten (F1...F12) ist abhängig von Dienst- und Anwenderprogrammen. Es gibt leider keinen Standard für die Belegung der Funktionstasten durch Anwenderprogramme. Einige wenige Übereinstimmungen sind aber in den meisten Fällen vorhanden. Beispielsweise löst F1 bei den meisten Programmen eine Hilfsfunktion aus.
Im Betriebssystem werden nur zwei Tasten sinnvoll verwendet:
<F1> kopiert jeweils ein Zeichen des letzten Befehls in die Eingabezeile; so kann Zeichen für Zeichen der letzte Befehl hervorgeholt werden (z. B. bis zu dem Buchstaben, bei dem man sich verschrieben hat).
<F3> kopiert das letzte Kommando vollständig wieder in die Eingabezeile (z. B. wenn mehrmals der gleiche Befehl ausgeführt wird oder korrigiert werden soll).
Das gleichzeitige Betätigen der Tasten **<CTRL> <ALT> <DEL>** führt zum erneuten Laden des Betriebssystems (Warmstart). Es kann notwendig werden, wenn das Betriebssystem zum Beispiel durch unsaubere Programmabläufe seine Funktion nicht mehr erfüllt (Absturz).

Ein Blick auf die Systemdiskette

Auf jedem Datenträger ist ein Inhaltsverzeichnis (Directory) über alle vorhandenen Daten angelegt. Mit dem folgenden Befehl kann man sich dieses auf dem Bildschirm anzeigen lassen.
A>dir <ENTER>

Datentraeger in Laufwerk A ist Dos 3.30
Inhaltsverzeichnis von A:
COMMAND COM 2607 11.11.87 11.11
1 File(s) xxxxxx byte free

Bei der Eingabe von Befehlen können Groß- und Kleinbuchstaben gleichermaßen verwendet werden.
Der **DIR**-Befehl listet uns die Namen aller Dateien auf dem aktuellen Laufwerk. Ein Name besteht aus bis zu acht Zeichen. Diese dürfen beliebig sein, aber folgende Zeichen sind nicht erlaubt:
/ \ [ ] : ; , . = | " < > +
Warum der Doppelpunkt nicht erlaubt ist, wissen wir bereits. Auch die Zeichen ? und * haben eine andere Bedeutung, wie wir noch sehen werden. Ist der Dateiname kürzer als 8 Zeichen, so wird automatisch mit Leerzeichen aufgefüllt. Es folgen – durch einen Punkt getrennt – drei Zeichen, die den Typ der Datei kennzeichnen. Für die eigenen Dateien kann diese Erweiterung (Extention) frei (aber sinnfällig) gewählt werden, zum Beispiel
.TXT für allgemeine Textdateien
.DOC für Dokumentationen
.PAS für Pascal-Quelltexte
.ASM für Assembler-Quelltexte
usw.
Daneben gibt es Erweiterungen, die reserviert sind. Dazu gehören
.EXE (execute), ausführbare Programme im DOS-Hausformat
.COM (command), ausführbare Programme mit CP/M-Konventionen
.BAT (batch), Stapelprogramme, die einfache DOS-Befehle enthalten, wie sie auf Kommandoebene verwendet werden können
.$$$ temporäre Datei, wie sie z. B. Wordstar anlegt. Bei korrekter Arbeit dürfte sie nicht auf der Diskette vorzufinden sein, aber wenn während der Arbeit ein RESET erfolgt (Stromausfall oder RESET-Knopf) kann das schon mal vorkommen. Kann gelöscht werden.
.BAK Sicherungsdatei; die meisten Editoren bewahren die vorletzte Version unter diesem Namen zur Sicherheit auf.
Im allgemeinen ist an allen Stellen, wo ein Dateinamen gefordert wird, zusätzlich die Angabe eines Laufwerks zulässig.
Der DIR-Befehl präsentiert uns außer dem Dateiname noch die Größe der Datei, das Datum und die Uhrzeit der letzten Änderung. Beim Schreiben einer Datei (also auch beim Ändern) trägt DOS die Zeit automatisch ein.
Auf der Systemdiskette befinden sich neben den gezeigten noch zwei weitere Dateien, die DOS nicht anzeigt: IO.SYS (bzw. IBMBIO-.COM bei IBM) und MSDOS.SYS (bzw. IBM-DOS.COM). Sie besitzen die Attribute einer verborgenen Datei. Das hat nicht den Grund, diese Dateien vor irgend jemandem zu verstecken (schließlich haben wir unser Betriebssystem ja ehrlich gekauft), sondern soll den Bildschirm von ihnen frei und übersichtlich halten. Diese Dateien und die Datei COMMAND.COM gehören zum Betriebssystem. Sie müssen auf der Systemdiskette

```
Laden eines zweiten Kommandointerpreters

COMMAND [lw:] [pfad] [device] [/e:nnnn]
        [/p] [/d] [/f] [/c befehl]

lw:         Laufwerk
pfad        Zugriffspfad
device      legt das Standard-Ein-/Aus-
            gabegerät fest (Standard: CON)
/e:nnnn     Größe des Environments
            (Standard: 160 Byte
            Maximum: 32 KByte)
/p          installiert COMMAND resident,
            d.h. er kann nicht verlassen
            werden
/d          schaltet die Abfrage von Datum
            und Zeit ab
/f          unterdrückt der Rückfrage bei
            Diskettenfehlern
/c befehl   veranlaßt COMMAND zur Aus-
            führung eines Befehls und
            Rückkehr in das aufrufende
            Programm
```

(oder -platte) vorhanden sein. Während die ersten beiden Dateien (die den Betriebssystemkern enthalten) nur zum Systemstart benötigt werden, muß COMMAND.COM immer auf dem Startlaufwerk präsent sein. Diese Datei ist der Kommandointerpreter (oder auch Shell), der alle wichtigen Befehle der DOS-Kommandooberfläche – also zum Beispiel dir – enthält. Wird ein Programm geladen (beispielsweise WordStar oder dBase), so wird er nicht benötigt und deshalb überspeichert. Nach Verlassen des Programms lädt DOS ihn vom Startlaufwerk nach. Allein deshalb schon ist es ratsam, das System von der Festplatte zu laden. Für den Fall, daß nur ein Laufwerk zur Verfügung steht, erlaubt DOS einen Diskettenwechsel zum Nachladen des Kommandointerpreters.
COMMAND.COM ist ein Programm wie jedes andere. Wir können es mit
command <ENTER>
starten und würden uns jetzt im zweiten Kommandointerpreter befinden. Davon ist nichts zu merken (denn beide sind identisch). Verlassen wir ihn also mit
exit <ENTER>
wieder. Die Möglichkeit des Ladens eines zweiten Kommandointerpreters kann durchaus sinnvoll sein, etwa aus einem Programm heraus oder um ein komfortableres Modell zu nutzen.
Bleiben wir noch einen Moment beim DIR-Befehl. Soll nicht das aktuelle Laufwerk angezeigt werden, so kann ein Parameter angegeben werden, zum Beispiel
dir c: <ENTER>
für das Laufwerk C:. Leerzeichen sind stets dort nötig, wo sie in den Beispielen vorgegeben sind, in diesem Falle zwischen dir und c:. Eine weitere Möglichkeit ist das Listen einer Gruppe von Files. Haben Sie zum Beispiel eine Textdatei geschrieben, deren Namen Sie vergessen haben (und warum sollte es Ihnen anders gehen als mir), und die Diskette ist voller Dateien: Hier wäre es möglich, sich alle Dateien vom Typ TXT auflisten zu lassen, und zwar mit
dir *.txt <ENTER>
Der Stern ist ein sogenannter Joker (oder Wildcard), ebenso wie das Fragezeichen. Ein Stern steht für einen beliebigen Namen oder eine beliebige Erweiterung (bzw. deren Rest), während das Fragezeichen für einen einzelnen beliebigen Buchstaben eingesetzt werden kann.
Einige Beispiele:

a*.*	alle Dateien, die mit a beginnen
pit.*	alle Dateien für Pit
M??er.txt	alle Textdateien für beliebig geschriebene Meiers
***.do?**	alle Dokumentationen, egal ob doc oder dok
.	alle Dateien

Die Joker haben nicht nur in Verbindung mit dem DIR-Befehl Bedeutung, sondern können überall verwendet werden, wo keine eindeutigen Dateibezeichnungen gefordert sind. Trotzdem darf man beim Löschen getrost etwas vorsichtig sein. Wer das Inhaltsverzeichnis seiner Diskette schwarz auf weiß mit nach Hause nehmen will, kann auch den Drucker zuschalten, der die Bildschirmanzeige mitdruckt.

Beispiel:
<CTRL> <P>
dir
<CTRL> <P>
Befinden sich sehr viele Dateien im Verzeichnis, so paßt die Anzeige nicht auf den Bildschirm. Dafür hält der DIR-Befehl zwei Schalter bereit:

dir/p Anzeige seitenweise (für page), nach 23 Zeilen wird angehalten und auf Tastendruck gewartet.
dir/w Ausgabe in fünf Spalten, dafür entfallen Dateigröße und Datum.

Das Dateiverzeichnis

Wer schon an 8-Bit-Technik gearbeitet hat und an dem heillosen Durcheinander auf den Disketten fast verzweifelt ist, der wird jetzt die Möglichkeit unter MS-DOS gerne nutzen, alle Dateien nach bestimmten Kriterien zu ordnen. Unter DOS gibt es nicht nur ein Verzeichnis, sondern ein Hauptverzeichnis (Rootdirectory) und beliebig viele Unterverzeichnisse (Subdirectories). In Bild 1 ist eine typische Verzeichnisstruktur dargestellt. Auf Disketten werden Sie nicht unbedingt Unterverzeichnisse finden, da der Platz begrenzt ist, aber auf der Festplatte sind sie ein Muß. Die Festplatte aus Bild 1 würde bei einem DIR folgendermaßen dargestellt werden:

```
Datentraeger in Laufwerk C ist ROGER
Inhaltsverzeichnis von C:\

TOOLS         <DIR>   15.01.89  08.15
TEXTE         <DIR>   18.06.89  10.00
DOS           <DIR>   19.10.89  14.00
COMMAND COM   2607    28.02.89  17.30
          1 File(s) xxxxxx byte free
```

Angezeigt werden nur die Dateien im Hauptverzeichnis und dessen direkte Unterverzeichnisse. Diese werden durch <DIR> kenntlich gemacht. Um in das Verzeichnis TEXTE zu wechseln, wird der Befehl **CD** oder **CHDIR** (für Change Directory) verwendet:
cd texte <ENTER>
dir <ENTER>

```
Datentraeger in Laufwerk C ist ROGER
Inhaltsverzeichnis von C:\TEXTE
.          <DIR>  18.06.89  10.00
..         <DIR>  01.01.80  00.00
BRIEFE     <DIR>  15.01.89  08.15
BERICHTE   <DIR>  18.06.89  10.00
          0 File(s) xxxxxx byte free
```

Neben den beiden neuen Unterverzeichnissen (vergleiche Bild 1) werden noch zwei weitere Directories gefunden, die die Namen **.** und **..** haben. Dabei ist **.** das Verzeichnis, in dem wir uns gerade befinden (was an dieser Stelle ohne Belang ist) und **..** das übergeordnete Verzeichnis. So würde uns ein
cd ..
wieder ins Hauptverzeichnis zurückführen. Auf diese Weise kann man sich durch den gesamten Verzeichnisbaum bewegen. Um sich nicht von Ast zu Ast hangeln müssen, kann das Zielverzeichnis auch vollständig angegeben werden – unabhängig davon, wo man sich gerade befindet. Die Verzeichnisse werden durch ein \ (Backslash) getrennt; für das Wurzelverzeichnis steht nur ein \. Verwechseln Sie dieses Zeichen nicht mit dem Schrägstrich / (auch Slash genannt), wie er zum Beispiel für die Schalter des DIR-Befehls benutzt wird. Für das Abtrennen von Verzeichnissen kann dieser zwar in der Tat alternativ verwendet werden, aber zur Kennzeichnung des Hauptverzeichnisses ist nur der Backslash zulässig. Aufgrund der ungünstigen Dreifachbelegung der Tasten ist dieses Zeichen bei den meisten Tastaturen nur durch gleichzeitiges Betätigen der Tasten <CTRL> <ALT> und der entspechend gekennzeichneten Taste auf den Bildschirm zu bringen. Wie bereits beschrieben, können wir aber auch das Wissen ausnutzen, daß der Backslash als 92 (dezimal) kodiert wird. Durch Drücken der <ALT>-Taste und Eintippen von 92 auf dem rechten Ziffernblock. Beim Loslassen der <ALT>-Taste erscheint ein \.
Von jedem Punkt der Dateihierarchie kommt man mit
**cd **
ins Hauptverzeichnis und mit
cd \texte\briefe\liebe\susi
ins Verzeichnis Susi. Diese Angabe des Weges wird als *Pfad* bezeichnet. Grundsätzlich ist überall dort, wo ein Dateiname einzugeben ist, zusätzlich die Angabe eines Pfades möglich.

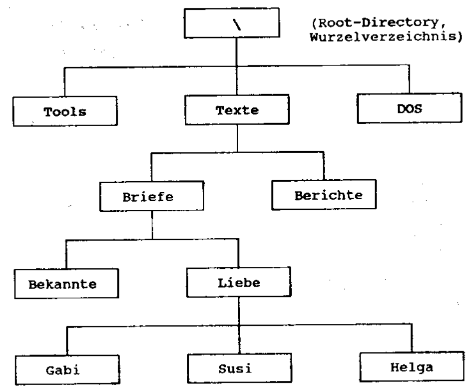


Bild 1 Beispiel für ein hierarchisches Dateiverzeichnis

```
Anzeigen Dateiverzeichnis

DIR [lw:] [pfad] [dateiname] [/p] [/w]

lw:        Laufwerk (z.B. A: oder C:)
pfad       Zugriffspfad (z.B. /texte/briefe)
dateiname  Dateibezeichnung
           (Joker '*' und '?' sind erlaubt)
/p         Pause nach Anzeige einer Seite
/w         Anzeige in Kurzform

Wechseln Unterverzeichnis

CD [lw:] [pfad]            oder
CHDIR [lw:] [pfad]

lw:        Laufwerk
pfad       Zugriffspfad ('..' für das über-
           geordnete Verzeichnis)

Anlegen eines neuen Unterverzeichnisses

MD [lw:] [pfad]            oder
MKDIR [lw:] [pfad]

lw:        Laufwerk
pfad       neu anzulegendes Verzeichnis

Löschen Unterverzeichnis

RD [lw:] [pfad]            oder
RMDIR [lw:] [pfad]

lw:        Laufwerk
pfad       zu löschendes Verzeichnis
           (muß leer sein)
```

Das Anlegen eines neuen Verzeichnisses ist mit dem Befehl **MD** oder **MKDIR** (für Make Directory) möglich. Um also die Briefe an unsere neue Errungenschaft richtig abzulegen, müssen wir uns mit
cd \texte\briefe\liebe
in das übergeordnete Verzeichnis bewegen und mit
md maria
ein neues Unterverzeichnis anlegen. Als Alternative steht die Angabe des gesamten Pfades:
md \texte\briefe\liebe\maria
Verzeichnisse können gelöscht werden, wenn sie keine Dateien und auch keine Unterverzeichnisse enthalten. Vom Löschen ebenfalls ausgenommen sind das Haupt- und das aktuelle Verzeichnis. Der Befehl dafür lautet **RD** oder **RMDIR** (für Remove Directory). Als Beispiel löschen wir das Verzeichnis, daß sich am ehesten entbehren läßt:
rd \texte\berichte

Weitere wichtige Befehle

Datei löschen
Mit dem **DEL**-Befehl können Dateien gelöscht werden. Als Parameter muß stets die Datei oder Dateigruppe angegeben werden, die gelöscht werden soll.
Beispiele:
del erfolg.txt
del \texte\briefe\liebe\susi\maerz.txt
del *.*
Die letzte Kommandozeile löscht alle Dateien des aktuellen Verzeichnisses. Bei der Verwendung der Joker beim Löschen ist Vorsicht geboten, oftmals kann man nicht übersehen, welche Dateien dazugehören. Die meisten Systeme akzeptieren statt **DEL** auch **ERASE**, was den CP/M-Nutzern entgegenkommt.

Datei umbenennen
Zum Umbenennen steht der Befehl **REN** oder **RENAME** zur Verfügung. Im Gegensatz zu CP/M erwartet DOS zuerst den alten und dann den neuen Namen.
Beispiel:
ren alt.txt neu.txt

Eingabe von Datum und Zeit
Zur Anzeige und zur Eingabe von Datum und Zeit dienen die Befehle DATE und TIME. Auf die Eingabe von **TIME** erscheint
Uhrzeit: 16:14:59.99
Neue Uhrzeit:
Die Anzeige kann mit <ENTER> quittiert werden, und wir wissen, daß in kürze Feierabend ist. Alternativ kann eine neue Zeit eingegeben werden, aber das sollten Sie ja schon am Anfang tun. Soll auf jeden Fall eine neue Zeit eingestellt werden, so kann das direkt geschehen:
time 16:30
Der **DATE**-Befehl funktioniert in gleicher Weise.

Kopieren

Einer der wichtigsten Befehle – wenn nicht gar der wichtigste überhaupt – ist der **COPY**-Befehl. Er wird in der Form
copy quelle ziel
verwendet und bietet eine Vielzahl von verschiedenen Anwendungsmöglichkeiten. Dabei stehen Quelle und Ziel für eine Datei oder eine Gruppe von Dateien (evtl. mit Laufwerk und Pfad) oder auch für ein Gerät. Als Ziel ist die Angabe eines Laufwerkes oder Verzeichnisses ausreichend.
Beispiel:
copy antrag.txt a: kopiert die Datei antrag.txt aus dem aktuellen Verzeichnis mit gleichem Namen nach a: Möglich ist auch das Kopieren innerhalb eines Laufwerkes in ein anderes Verzeichnis. Auf den Zielparameter kann ganz verzichtet werden, wenn es sich um das aktuelle Verzeichnis des aktuellen Laufwerkes handelt. Als Name der Zieldatei wird der Name der Quelldatei angenommen.
Beispiel:
copy a:\praemie.txt kopiert praemie.txt aus dem Wurzelverzeichnis des Laufwerkes a: ins aktuelle Verzeichnis des aktuellen Laufwerkes. Ist als Zielparameter auch ein Filename angegeben, so wird die Datei beim Kopieren umbenannt. Werden beim Zielnamen keine Joker verwendet, so muß auch der Quellname eindeutig sein.
Beispiel:
copy a:alt.txt b:neu.txt
Da es sich um einen Kopiervorgang handelt, bleibt die alte Datei selbstverständlich erhalten. Durch Umbenennen während des Kopiervorganges ist es möglich, eine Datei in dasselbe Verzeichnis zu kopieren. Auch die Joker können in vielfältiger Weise genutzt werden.
Beispiel:
copy *.txt a: kopiert alle Texdateien nach a:
Ein wichtiges Merkmal von DOS (auf das an dieser Stelle nicht näher eingegangen werden kann) ist, daß alle Geräte wie Textdateien behandelt werden, so daß es möglich ist, eine Datei auf den Bildschirm oder den Drucker zu kopieren. Zur besseren Unterscheidung werden die Geräte im folgenden groß geschrieben. Die wichtigsten DOS-Geräte sind:

CON	Konsole, beinhaltet Bildschirm und Tastatur
PRN	Drucker
COM1	erste serielle Schnittstelle

```
Datei(en) löschen

DEL [lw:] [pfad] [dateiname]          oder
ERASE [lw:] [pfad] [dateiname]

lw:          Laufwerk
pfad         Zugriffspfad
dateiname    Dateibezeichnung (Vorsicht bei
             Verwendung der Joker '*' und '?')
```

```
Datei(en) umbenennen

REN [lw:] [pfad]  datei_alt  datei_neu    oder
RENAME [lw:] [pfad]  datei_alt  datei_neu

lw:          Laufwerk
pfad         Zugriffspfad
             Laufwerk und Pfad werden nur
             für den alten Namen angegeben
datei_alt    alte Dateibezeichnung
datei_neu    neue Dateibezeichnung
```

```
Anzeige von Textdateien

TYPE [lw:] [pfad] dateiname

lw:          Laufwerk
pfad         Zugriffspfad
dateiname    Dateibezeichnung
```

```
Uhrzeit anzeigen/einstellen

TIME [hh[:mm[:ss[.xx]]]]

hh   Stunden    (0 .. 23)
mm   Minuten    (0 .. 59)
ss   Sekunden   (0 .. 59)
xx   Hundertstel (0 .. 99)
```

```
Datum anzeigen/einstellen

DATE [tt.mm.[jj]jj]

tt       Tag   (1 .. 31)
mm       Monat  (1 .. 12)
[jj]jj   Jahr (mit Jahrhundert
         oder ohne)
```

```
Anzeige MS-DOS Version

VER
```

```
Namen des Datenträgers anzeigen

VOL [lw:]

lw:    Laufwerk
```

```
Bildschirm löschen

CLS
```

```
Prüfen von geschriebenen Daten

VERIFY [on/off]

on     einschalten
off    ausschalten
```

```
Datei(en) kopieren

COPY [lw:] [pfad] dateiname [+[lw:]dateiname[+..]]
     [/a] [/b] [lw:] [pfad] [dateiname] [/a] [/b] [/v]

lw:          Laufwerk
pfad         Zugriffspfad
dateiname    Dateibezeichnung (vielfältige Verwendung
             der Joker möglich)
/a           Text-(ASCII-) Dateien
             (Ctrl-Z als Dateiendekennzeichen findet
             Beachtung)
/b           Programm-(Binär-) Dateien
             (Ctrl-Z wird nicht als Dateiendekennung
             betrachtet)
/v           (verify) Kontroll-Lesen nach dem Schreiben
```

Der Drucker kann nur als Ausgabegerät genutzt werden, die anderen Geräte zur Ein- und Ausgabe.
Einige Beispiele:
copy CON kleiner.txt Mini-Editor; alle Eingaben der Tastatur werden in die Datei geschrieben, Abschluß mit <CTRL> <Z> ENTER
copy CON PRN Schreibmaschine; die Ausgabe erfolgt auf den Drucker.
copy trans.txt COM1 kopiert den Text zur seriellen Schnittstelle, hoffentlich wird er dort auch aufgefangen.
Eine manchmal recht nützliche Möglichkeit des COPY-Befehls ist das Verketten von Dateien.
Beispiel:
copy eins.txt+zwei.txt
Soviel zu den vielfältigen Möglichkeiten des COPY-Befehls. Ein wenig Vorsicht ist geboten, da bereits existierende Zieldateien gleichen Namens ohne Warnung überschrieben werden. Ein Wunsch, den uns COPY nicht erfüllt, ist das Kopieren ganzer Verzeichnisse oder ganzer Disketten. Deshalb werden wir später Werkzeuge kennenlernen, die das unterstützen.

wird fortgesetzt

Fortsetzung von Seite 110

(1024 × 768 ohne Zeilensprung) wie der Eizo 9070 arbeiten nicht mehr unterhalb 20-kHz-Zeilenfrequenz. An einer CGA- oder Hercules-Karte sind solche Geräte unbrauchbar, nicht aber bei entsprechenden Emulationen einer VGA-Karte, die dann mit mehr als 30-kHz-Zeilenfrequenz arbeitet.

8514A

Den Grafikadapter 8514A mit 1024 × 768 Bildpunkten hat IBM nur für seine PS/2-Modelle mit Mikrokanal entwickelt. Hardwaremäßig verkoppelt steuern VGA auf dem Motherboard und 8514A gemeinsam den hochauflösenden 16-Zoll-Monitor 8514. Demgemäß bieten alle Nachbauer ihre 8514A-kompatiblen Karten mit integrierter VGA an, wenn die Karte für den Standard-AT-Bus gedacht ist.

Der 8514A wird über I/O-Ports mit „Zeichenkommandos" für seinen Grafikprozessor angesteuert, ist also keine Super-VGA, und belegt keinen Bildspeicher im Adreßraum der CPU. Zwar ermöglicht er schnelle Bildaufbauzeiten, die Darstellung auf dem über 4000 DM kostenden 8514-Schirm mit Zeilensprungverfahren ist aber wenig profihaft.

Mitbewerber, die auf den Profibereich zielen, bieten daher Lösungen ohne Zeilensprung an. Vor allem Karten mit den Grafikprozessoren 34010/34020 von Texas Instruments finden wegen der Softwareschnittstelle TIGA im Hochleistungsbereich (CAD, DTP) viel Anklang. Da könnte sich ein wirklicher Profistandard etablieren. Andere setzen auf absolute Hardware-Kompatibilität zum 8514A, weil Microsoft angeblich beim Presentation Manager des OS/2 an der von IBM für den 8514A verordneten Softwareschnittstelle vorbeiprogrammiert. Es dürfte also bald eine Spaltung in hardware- und softwarekompatible Nachbauten geben.

Augen auf beim Kauf

Ein Monitor ist immer nur so gut wie seine Endkontrolle. Bestes Material und aufwendige Auslegung werden durch einen schlechten Abgleich von Schärfe und Konvergenz hinfällig. Die Gefahr, an ein „Montagsgerät" zu gelangen, ist bei Markengeräten erfahrungsgemäß am geringsten. Der Kauf eines No-name-Produkts ist deshalb aber keineswegs bedenklich, man muß es nur sehr genau im Betrieb begutachten dürfen, und zwar genau das Exemplar, das man zu kaufen beabsichtigt. Es ist nicht damit getan, ein perfektes Vorführgerät zu bestaunen, um dann ein ungeprüftes Gerät aus dem Lager mitzunehmen.

PC-Grafikadapter und passende Monitore

Karte	Monitor
CGA-Karte	BAS-Monitor TTL-CGA-Monitor (AGA) RGB-Monitor EGA-Monitor (normgerecht nach IBM) Multiscan bis max. Auflösung 800 × 600
Hercules	TTL-Monitor TTL-CGA-Monitor (AGA) Multiscan bis max. Auflösung 800 × 600
AGA-Karte	CGA-Monitor (nur im CGA-Modus) TTL-Monochrom (nur im MDA-/MGA-Modus) Multiscan bis max. Auflösung 800 × 600
EGA-Karte	EGA-Monitor Multiscan bis max. Auflösung 800 × 600
Super-EGA	Multiscan bis max. Auflösung 800 × 600
VGA-Karte	VGA-Monitor Analog-Multiscan bis max. 1024 × 768
Super-VGA	Analog-Multiscan bis max. 1024 × 768

Bei Farbmonitoren ist der Abgleich der Konvergenz, also der Deckungsgleichheit des roten, blauen und grünen Teilbildes, ein wichtiges Gütemerkmal. Nur bei teuren Spezial-Monitoren darf man absolut fehlerfreie Konvergenz erwarten. Bei VGA- und Multiscan-Monitoren der unteren und mittleren Preisklasse (14-Zoll-Geräte bis 1500 DM) gibt es in der Regel immer sichtbare Farbränder in ein oder zwei Ecken des Bildschirms. Diese sollten aber in der Größenordnung von zwei bis drei Millimetern liegen und sich auf kleine Bereiche beschränken. Konvergenzfehler von mehr als drei Millimetern, die sich über weite Bereiche erstrecken, sind ein Grund zur Reklamation. Das gilt auch für Linearitätsfehler, wenn also etwa einzelne Zeichen innerhalb einer Zeile und in verschiedenen Zeilen nicht gleich groß erscheinen.

Eine schnelle und leicht zu realisierende Kontrolle ist daher ein vollgeschriebener Textbildschirm mit 25 Zeilen zu 80 Zeichen. Die Zeichen sollten gut lesbar und frei von Verzerrungen und Stauchungen sein. Farbränder dürfen sich nur in ein oder zwei Eckbereichen zeigen.

Einen sichtbaren Zeilenrücklauf, der schon mal bei billigen TTL-Monitoren vorkommt, darf man ebenfalls nicht klaglos hinnehmen. Bei den inzwischen recht beliebten Schwarzweiß-Multiscans kann als Gütemerkmal gelten, wie stabil die Bildschärfe ist. Bei Geräten minderer Qualität sinkt die Schärfe beim Anheben von Helligkeit und Kontrast rasch ab.

Monitore dürfen im normalen Betrieb auch nicht „wabern", so daß Schriftzüge an einigen Stellen des Schirms auf und ab tanzen. Solch rhythmische Störungen können mal entstehen, wenn zwei Monitore nebeneinander stehen, ab einem Meter Abstand zu anderen Geräten muß aber absolute Ruhe einkehren.

/1/ D. Grell, D. Hoffmann, Bilder machen sie alle, 16 VGA-Karten mit erhöhter Auflösung im Vergleich, c't 5/89, S. 80 ff.

/2/ Manfred Bertuch, Bunte Alleskönner, 10 Multiscan-Monitore im Vergleich, c't 4/89, S. 52 ff.

Nachdruck aus c't – Magazin für Computertechnik, Heft 1/1990

Teil 3 dieses Beitrages erscheint in MP 5/1990 und befaßt sich mit Festplatten- und Druckerproblemen.

Anschluß gewinnen

COM3 und COM4 im PC nutzen

Frank Schmidt

Wie ein Gerücht kursiert unter den PC-Usern die Information, DOS könne auch mit den seriellen Schnittstellen COM3 und COM4 umgehen. Wer den Wahrheitsgehalt dieser Aussage genauer überprüft, wird jedoch meistens enttäuscht, weil viele Rechner von den zusätzlichen Schnittstellen nichts wissen wollen. Doch es stimmt: DOS bedient auch COM3 und COM4, wenn man ihm mitteilt, wo sie sich befinden.

Wer schon einmal versucht hat, im PC die seriellen Ports COM3 und COM4 anzusprechen, der kennt das Problem: DOS (bis zur neuesten Version) behandelt diese beiden Schnittstellen stiefmütterlich bis gar nicht, aber auch Anwender-Programme unterstützen sie nur sehr selten. Was ist also zu tun, wenn beispielsweise COM1 und COM2 von einer Maus und einem Modem belegt sind, man aber außerdem noch einen seriellen Drucker oder Plotter anschließen möchte?

Beim Blick auf die reservierten I/O-Adressen im PC stellt sich heraus, daß dort Platz für bis zu vier serielle Schnittstellen-Bausteine des Typs 8250 beziehungsweise 16450 vorhanden ist. Da die meisten der zur Zeit erhältlichen Schnittstellen-Karten so konfigurierbar sind, daß sie auch an den für COM3 und COM4 reservierten Adressen arbeiten können, sollte man meinen, es sei ja alles kein Problem: Karte richtig jumpern, in den Rechner einstecken und los geht's.

Schon beim Versuch, mit dem DOS-Programm MODE die Parameter der Schnittstelle einzustellen, erleidet man Schiffbruch: MODE kann nur die Schnittstellen COM1 und COM2 bedienen. Versucht man, die Maus zum Beispiel an COM3 oder COM4 anzuschließen, so wird man wiederum enttäuscht: Auch der Maustreiber verweigert die Zusammenarbeit mit diesen beiden Schnittstellen.

Nach längerem Herumprobieren kommt man deshalb sehr schnell auf die Idee, daß der Kauf der neuen Schnittstellen-Karte wohl eine Fehlinvestition sei und findet sich damit ab, daß man in Zukunft Drucker und Modem abwechselnd am gleichen Port betreibt.

	I/O-Adresse	BIOS-RAM
COM1	3F8h-3FFh	0040h:0000h
COM2	2F8h-2FFh	0040h:0002h
COM3	3E8h-3EFh	0040h:0004h
COM4	2E8h-2EFh	0040h:0006h
LPT1	378h-37Fh	0040h:0008h
LPT2	278h-27Fh	0040h:000Ah
		0040h:000Ch
		0040h:000Eh
Paralleler Port auf der Monochrom-Karte: 3BC-3BEh		

Im I/O-Adreßbereich des PC ist die Lage von vier seriellen und drei parallelen Ports definiert, deren Basisadressen legt das BIOS im RAM ab Adresse 0040h:0000 ab. Im allgemeinen fehlen die Adressen für COM1 und COM2; eine 0 anstelle der Adresse bedeutet, daß der zugehörige Schnittstellen-Baustein nicht existiert.

Interrupt 14h: Serielle Schnittstelle

AH=0: Schnittstelle initialisieren
DX: Schnittstellen-Nr. (0=COM1)
AL: Bits 0.1 Wortlänge
Bit 2 Anzahl Stopbits
Bits 3, 4 Parität
Bits 5–7 Bitrate
Rückgabe-Werte
AH: Schnittstellen-Status
AL: Modem-Status

AH=1: Zeichen senden
DX: Schnittstellen-Nr.
AL: Zeichen
Rückgabe-Werte
AH: Schnittstellen-Status

AH=2: Zeichen empfangen
DX: Schnittstellen-Nr.
Rückgabe-Werte
AL: Zeichen
AH: Input-Status

AH=3: Zeichen abfragen
DX: Schnittstellen-Nr.
Rückgabe-Werte
AH: Output-Status
AL: Modem-Status

Schnittstellen-Status-Byte

Bit 0: Zeichen wurde empfangen
Bit 1: Overrun-Error
Bit 2: Paritätsfehler
Bit 3: Protokoll nicht eingehalten
Bit 4: Unterbrechungs-Signal empfangen
Bit 5: Transmission-Hold-Register leer
Bit 6: Transmission-Shift-Register leer
Bil 7: Time-Out-Fehler

Über den Interrupt 14h kann man auf die Funktionen des ROM-BIOS zur Bedienung der seriellen Schnittstellen zurückgreifen. Es gibt keine Beschränkung auf COM1 und COM2.

```
comment %

 FINDCOM 2.0, c't / Frank Schmidt
   (MASM 4.0/5.0, OPTASM, Turbo-Assembler)

 Prüft die Existenz der serielllen Schnittstellen
 COMi bis COM4 und trägt ihre Adressen in's BIOS-RAM ein
 Übersetzen mit      MASM FINDCOM;
                     LINK FINDCOM;
                     EXE2BIN FINDCOM.EXE FINDCOM.COM
%

;******* Definition der Konstanten und Adressen **********

; Interrupts
        msdos      = 21h
        PrtStrg    = 09h    ; Zeichenkette ausgeben
        RetDos     = 4Ch    ; Rückkehr zum DOS (Programmende)

; Konstanten
        ComMax     = 4      ; maximale Nummer des COM-Kanals
        cr         = 13     ; Wagenrücklauf
        lf         = 10     ; Zeilenvorschub
        etx        = '$'    ; Markierung String-Ende

; Adressen
        BiosVar    = 0040h ; BIOS-Variablen Segment
        ComPtr     = 0000h ; Adressen der COM-Ports

;***************** Macro-Definitionen ********************

; dos: INT-21h-Aufruf mit Funktionsnummer als Parameter

dos     macro   FctnNo
        mov     ah,FctnNo
        int     msdos
        endm

; Print: String ausdrucken, Adresse als Parameter

Print   macro   StrAdr
        mov     dx,offset StrAdr ;; Adresse des Strings
        dos     PrtStrg
        endm

;********** Segment-Definitionen für COM-Datei ***********

cseg    segment 'code'

        assume cs©seg, ds©seg, es©seg, ss©seg

        org     0100h  ; Beginn für COM-Datei
;***************** Programm-Beginn ***********************

FindCom:        jmp start

signon: db      cr,lf,'FINDCOM vers 2.0, c''t / Frank'
        db      ' Schmidt 1989',cr,lf,etx

start:
        Print   signon      ; Signon-Meldung ausgeben
        call    Install     ; COM-Ports suchen und ihre
                            ; Adressen in's BIOS-RAM
        Print   ComPrtsMsg  ; Anzahl COM-Ports ausgeben
        dos     RetDos      ; Programmende

Install proc
        mov     bx,BiosVar
        mov     es,bx        ; ES zeigt auf BIOS-RAM
        mov     di,ComPtr   ; Zeiger auf BIOS-COM-Port-Adr.
        mov     si,0         ; Zeiger in Tabelle der Adr.
        mov     cx,8
        mov     bx,0

ClrComPtrs:
        mov     byte ptr es:[di][bx],0   ; Zuerst alle
        inc     bx   ; Adreß-Zeiger des BIOS zurücksetzen
        loop    ClrComPtrs

Search8250Loop:
        mov     dx,word ptr ComAddresses[si] ;COM-Adresse
        call    Check8250          ; untersuchen, ob 8250 da
        jnz     Next8250          ; falls nicht, dann weiter
        inc     byte ptr ComPorts   ; Port-Zähler erhöhen
        mov     es:[di],dx    ; BIOS die Adresse des Ports
                                               ; mitteilen
        add     di,2      ; Zeiger in BIOS-Bereich erhöhen

Next8250:
        add     si,2   ; Zeiger auf nächste mögl. Adresse
        cmp     si,2*ComMax        ; Alle Ports untersucht?
        jne     Search8250Loop  ; wenn nicht, dann weiter
        ret
Install endp

; Check8250: Überprüfen, ob ein serieller Schnittstellen-
; Baustein an einer bestimmten I/O-Adresse vorhanden ist
;
; Parameter:        dx = Basis-Adresse des 8250
; Rückgabewerte:  cl = 0: 8250 vorhanden, 0ffh: kein 8250
;                       zero-flag: gesetzt, falls 8250 da
; Zerstörte Register:  ax,flags

Check8250 proc
        mov     cl,0ffh        ; Default-Wert: kein 8250 da
        push    dx
        inc     dx
        inc     dx
        in      al,dx
        pop     dx
        test    al,0F8h
        jnz     EndCheck8250
        mov     cl,0   ; ok, sieht aus, als sei ein 8250 da
EndCheck8250:
        or      cl,cl     ; zero-flag entsprechend Ergebnis
        ret                                         ; setzen
Check8250 endp

;************* Strings und Konstanten *******************

ComPrtsMsg:     db 'Es wurden '
  ComPorts:     db '0 serielle Schnittstellen gefunden '
                db 'und installiert',cr,lf,etx

ComAddresses:   dw 03F8h, 02F8h, 03E8h, 02E8h

cseg    ends
        end     FindCom
```

Das Programm FINDCOM ermittelt bis zu vier serielle Schnittstellen und macht sie dem DOS bekannt. Die Versionen ab 3.3 können danach auch mit COM3 und COM4 umgehen.

Der Anfang allen Übels

Doch nicht jeder wird sich so schnell mit diesem Problem abfinden. Dem voll Tatendrang steckenden Programmierer kommt es gerade recht, wittert er doch eine Schwachstelle, die man „locker" mit einem kleinen Programm umschiffen könnte. Wo aber liegt diese Schwachstelle? In der Hardware jedenfalls nicht, das ist schon klar geworden. Das nächste Glied in der zu untersuchenden Kette ist das ROM-BIOS.

Für die Programmierung der seriellen Schnittstelle stellt das ROM-BIOS seine Dienste über den Interrupt 14h zur Verfügung. Mit Hilfe dieses Interrupts kann man die Schnittstelle initialisieren, Zeichen ausgeben, Zeichen einlesen und den Status der Schnittstelle abfragen. Beim Aufruf des Interrupts wird die Funktionsnummer wie üblich im Register AH übergeben. Das Register DX enthält die Nummer des zu programmierenden Ports: Von einer Einschränkung dieses Wertes auf 0 und 1 für COM1 und COM2 ist nichts zu entdecken. Im Gegenteil: COM3 und COM4 werden genauso gut bedient, wenn einige Bedingungen erfüllt sind.

Das ROM-BIOS geht davon aus, daß es zur Datenübertragung einen der eingangs erwähnten Bausteintypen mit entsprechendem Drumherum vorfindet. Es macht jedoch keine Einschränkungen bezüglich der Adressen der zu programmierenden Bausteine. Diese können sich beliebig im I/O-Bereich befinden. Wenn Sie ihre Schnittstellenkarte auf die I/O-Adresse 040h legen, macht's dem ROM-BIOS nichts aus – wenn man mal davon absieht, daß es zu Adreßkonflikten mit dem Timer-Baustein kommt, der sich bei jedem kompatiblen PC an dieser Stelle eingenistet hat.

Dennoch muß das ROM-BIOS wissen, an welcher Adresse der serielle Port liegt, den es als COM*soundsoviel* ansprechen soll. Zu diesem Zweck existiert im BIOS-Datenbereich ab Adresse 0040h:0000 eine Tabelle, in der die Adressen der seriellen und parallelen I/O-Ports abgelegt sind. Und hier ist wiederum Platz für die Basisadressen von vier seriellen Schnittstellen-Bausteinen. Das BIOS kann alle vier Schnittstellen bedienen – sobald ihre Adressen an der entsprechenden Stelle im BIOS-RAM eingetragen sind.

Lauter Nullen

Und das ist auch der erste Haken bei der Programmierung von COM3 und COM4. Schaut man sich den BIOS-RAM-Bereich mit Hilfe von DEBUG einmal genauer an, so stellt man fest, daß in den Adressen 0040h:0004 bis 0040h:0007, also dort, wo eigentlich die Adressen von COM3 und COM4 zu stehen hätten, meistens Nullen zu finden sind – unabhängig davon, wie viele Schnittstellen-Bausteine wirklich im System vorhanden sind. Das bedeutet für die Routine, die den Interrupt 14h bedient, daß diese Schnittstellen nicht vorhanden sind. Sie lassen sich folglich darüber auch nicht programmieren.

Für die Initialisierung der Tabelle mit den Adressen der Schnittstellen ist ebenfalls das ROM-BIOS zuständig. Es überprüft nach dem Einschalten des Rechners die beiden Adreß-Bereiche ab 3F8h und 2F8h auf das Vorhandensein der Interface-Chips und trägt deren Basis-Adressen in den BIOS-Datenbereich ein. Die meisten BIOS-Versionen beenden ihren Test, wenn sie zwei Schnittstellen gefunden haben und interessieren sich nicht weiter für COM3 und COM4. Allerdings gibt

es einige Ausnahmen, wie zum Beispiel das V20-BIOS, das die Adressen korrekt ermittelt und weiterreicht.

DOS 3.3 hat's gelernt

Falls das BIOS sich jedoch 'IBM-konform' verhält, braucht man ein kleines Zusatzprogramm, das die Adressen aller vorhandenen Interface-Chips korrekt ins BIOS-RAM einträgt. Für DOS-Versionen ab 3.30 ist das auch schon alles. Älteren DOS-Versionen jedoch sind die Ein-/Ausgabe-Geräte COM3 und COM4 völlig unbekannt. Das Programm MODE zum Beispiel verhält sich dort weiterhin stur. Und genauso steht es mit den anderen Teilen des Betriebssystems.
Bei diesen älteren Versionen hilft ein Trick. Man kann nämlich einen logischen Druckeranschluß (LPT) auf eine serielle Schnittstelle umleiten. So sorgt beispielsweise der Befehl

```
MODE LPT2=COM2
```

dafür, daß alle Zeichen, die an LPT2 gesandt werden, tatsächlich an COM2 ausgegeben werden. Das funktioniert natürlich nur für Drucker und ähnliche Geräte, die keine Daten zum Computer zurücksenden müssen. Ein Modem könnte man so nicht bedienen.
In Analogie dazu könnte man nun versucht sein, den Drucker an COM3 anzuschließen, dann via

```
MODE LPT2=COM3
```

den parallelen Port LPT2 auf den seriellen COM3 umzuleiten und schließlich den Drukker als LPT2 anzusprechen. Aber MODE wird den Parameter COM3 nicht akzeptieren. Außerdem müßte man noch irgendwie die Übertragungsparameter der Schnittstelle, an die der Drucker angeschlossen ist, einstellen können. Auch hier versagt MODE.
Kurz und gut – ein neues Programm würde das Problem auch hier lösen. Es sollte folgende Aufgaben erfüllen können: Zuerst muß es natürlich auch alle installierten COM-Ports finden und ihre Adressen in die Tabelle ab 0040h:0000 eintragen. Ferner muß es in der Lage sein, die Übertragungsparameter von COM1 bis COM4 einzustellen. Und zum guten Schluß muß es noch die logischen Geräte LPTx auf COM-Ausgänge umleiten können. Wir stellen hier ein kurzes Programm vor, welches die BIOS-Vektoren korrekt installiert.

Blick ins Programm

Im Programm selbst übernimmt die Prozedur 'Install' die eigentliche Arbeit. Sie löscht zuerst alle im BIOS-RAM eingetragenen COM-Adressen. In einer weiteren Schleife schaut es dann für alle vier Ports nach, ob ein Schnittstellen-Baustein vorhanden ist. Sollte das der Fall sein, so wird seine Adresse in das BIOS-RAM eingetragen.
Zu beachten ist, daß diese Routine die starre Verbindung zwischen den COM-Ports und den ihnen zugeordneten Adressen auflöst. Ein Beispiel: Falls Sie nur eine einzige Schnittstellenkarte haben, so können Sie die Adresse dieser Karte einstellen, wie immer Sie wollen, nach dem Start des Programmes ist sie dem Betriebssystem grundsätzlich als COM1 bekannt. Bei zwei Karten sind es COM1, COM2 und so weiter.
Interessant ist noch das Unterprogramm 'Check8250', in dem überprüft wird, ob sich ein serieller Schnittstellen-Baustein (8250 oder 16450) an einer bestimmten Adresse befindet. Die Routine findet sich in ähnlicher Form in nahezu jedem BIOS und wurde auf diversen Rechnern vom XT bis zur 386-PS/2-Maschine getestet. Einzig das Modell 30 von IBM lieferte eigenartige Ergebnisse, indem es einfach ein bis zwei Schnittstellen mehr als vorhanden meldete. In diesem Rechner läßt sich die Schnittstellen-Ermittlung nur mit Vorsicht einsetzen. Da es aber um die Installation von mehr als zwei Schnittstellen geht, dürften die Probleme nicht allzu groß sein. Bei vier installierten Schnittstellen arbeitet das Programm einwandfrei. Sollte es bei drei vorhandenen Schnittstellen vier melden, ist das auch nicht weiter tragisch, wenn Sie die vierte imaginäre Schnittstelle einfach nicht ansprechen.
Das Programm initialisiert die seriellen Schnittstellen nicht. Das ist aber kein Problem: Bevor Sie nämlich damit arbeiten, müssen Sie sowieso mit MODE die Übertragungsparameter einstellen – die Initialisierung wird dabei gleich mit erledigt.

Literatur
/1/ DOS 3.30 Technical Reference, IBM 1987
/2/ DOS 4.0 Technical Reference, IBM 1988
/3/ Michael Tischler, PC Intern – Systemprogrammierung, Data Becker 1988
/4/ Frank Brendle, Schnittstellen umbenennen, c't 9/87, S. 164
/5/ Hartung, Felsmann, Stiller, PC-Bausteine: Der UART 8250 als Tor zur seriellen Welt, c't 5/88, S. 204

Hinweis:
Die Satzanlage hat in Zeile 50 des Listings vier Copyright-Symbole eingeschmuggelt. Ersetzen Sie diese jeweils durch die Kombination :C.

Nachdruck aus c't – Magazin für Computertechnik, Heft 10/1989

Mikroprozessorsteuerungen und grafische Systeme

Dr. Reinhard Langmann, Berlin

Bisher wurden grafische Systeme innerhalb der Automatisierungstechnik vor allem für Leitrechner in der Prozeß- und Anlagenautomatisierung eingesetzt. Farbgrafiksysteme bilden jedoch zunehmend auch einen Schwerpunkt bei der Schaffung nutzergerechter Bedienoberflächen in Mikroprozessorsteuerungen für den prozeßnahen Bereich. Gefördert wird diese Entwicklung sowohl durch die Verfügbarkeit leistungsfähiger Grafikprozessoren als auch durch umfangreiche Aktivitäten zur Schaffung von grafischen Schnittstellen. Im folgenden Beitrag sollen einige Schwerpunkte der Einbeziehung grafischer Systeme in Mikroprozessorsteuerungen betrachtet werden.

Einführung

Der Einsatz von Farbgrafiksystemen zur Unterstützung der Mensch-Maschine-Kommunikation in hochleistungsfähigen Mikroprozessorsteuerungen bildet heute eine Voraussetzung, um die komplexen Steuerungsaufgaben im prozeßnahen Bereich erfolgreich lösen zu können. Besonders deutlich zeigt sich dieser Sachverhalt bei CNC-Steuerungen für Werkzeugmaschinen. Die steigende Funktionsvielfalt durch die Anwendung von 16-Bit- und 32-Bit-Steuerungen erzeugt neue Forderungen an die Mensch-Maschine-Schnittstelle. Dazu gehören beispielsweise

– die optimale grafische Gestaltung der Dialogabläufe für NC-Programmierung, Maschinenbedienung und Diagnose
– die grafische Kontrolle und die Überwachung von Steuerungs- und Prozeßvariablen
– die grafisch-dynamische Simulation des Bearbeitungsvorgangs bis hin zur grafischen Echtzeitsimulation komplizierter Fräsbearbeitungsprozesse.

Bild 1 zeigt eine Modellvorstellung für die Einbeziehung eines grafischen Mensch-Maschine-Interfaces in eine Mikroprozessorsteuerung. Über das Mensch-Maschine-Interface kommuniziert der Nutzer mit der Steuerung. Die Verbindung zwischen dem Steuerungskern und dem grafischen Mensch-Maschine-Interface erfolgt innerhalb der Steuerung über eine festgelegte Schnittstelle. Diese innere Schnittstelle überträgt in Richtung Mensch-Maschine-Interface komplexe Informationen über das Maschine-Prozeß-Verhalten und das Steuerungsverhalten sowie in Richtung Steuerungskern Bedienerkommandos für die Steuerung.
Die Spezifik der Einbeziehung eines grafischen Mensch-Maschine-Interfaces in eine Mikroprozessorsteuerung resultiert unmittel-

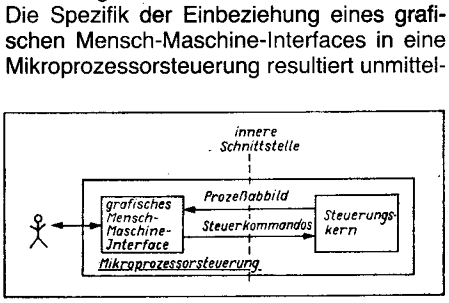


Bild 1 *Modellvorstellung für die Einbeziehung eines grafischen Mensch-Maschine-Interfaces in eine Mikroprozessorsteuerung*

bar aus den Besonderheiten des Einsatzes dieser Steuerungen für zeitkritische Steuerungs-, Regelungs- und Überwachungsaufgaben. Begrenzte ökonomische und technische Ressourcen sowie hohe Anforderungen an Betriebssicherheit und Zuverlässigkeit schränken den Handlungsspielraum für die Schaffung von steuerungstechnischen Lösungen im prozeßnahen Bereich ein. Grafische Systeme, die, eingebettet in Mikroprozessorsteuerungen, die Mensch-Maschine-Kommunikation unterstützen sollen, müssen deshalb unter anderem folgende Eigenschaften aufweisen:

– schnelle Visualisierung dynamischer Bildstrukturen
– Unterstützung der Parallelverarbeitung grafischer Applikationsdaten
– komfortable Window- und Viewporttechnik
– zuverlässige Funktion unter Echtzeitbedingungen
– geringer Speicheraufwand durch effiziente Bildstrukturen
– Portier- und Generierbarkeit.

Grafische Schnittstellen

Eingeführte grafische Systeme im Workstations- und zum Teil auch im PC-Bereich nutzen häufig standardisierte Schnittstellen für die grafische Programmierung, für grafische Geräteschnittstellen und für den Aufbau von Bilddateien. Dabei sollen folgende Zielstellungen erreicht werden:

– Portabilität der Anwendungsprogramme
– Bereitstellung eines anwendungsunabhängigen Funktionssatzes
– Unterstützung des Programmierers bei der Benutzung grafischer Methoden
– Unterstützung des Geräteherstellers bei der Auswahl sinnvoller Fähigkeiten der grafischen Geräte.

Im Bild 2 sind die typischen Schnittstellen eines grafischen Systems dargestellt.

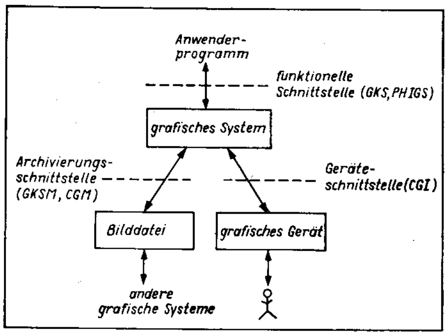


Bild 2 Schnittstellen eines grafischen Systems

Über die *funktionelle Schnittstelle* erfolgt die grafische Programmierung eines Anwenderprogramms. Weit verbreitet ist dafür das Grafische Kernsystem (GKS), das in seiner 2D-Variante seit 1985 als internationaler Standard anerkannt ist /1/. GKS ermöglicht bzw. unterstützt

– 2D-Linien- und Rastergrafik
– Aus- und Eingabe an mehreren grafischen Arbeitsplätzen
– temporäres Speichern grafischer Informationen sowie Bildstrukturierung mittels sogenannter Segmente
– permanentes Speichern grafischer Informationen in Bilddateien

– Hilfsmittel zur Anpassung des Anwenderprogramms an vorhandene Gerätefähigkeiten
– mehrere abwärtskompatible Implementierungsebenen mit zunehmenden funktionellen Fähigkeiten.

Die Leistungsfähigkeit des GKS kann stufenweise ausgebaut werden (Level-Konzept). Bild 3 verdeutlicht grob den Funktionsumfang von Level (Niveau) 0a bis Level 2c. Für in Mikroprozessorsteuerungen eingebettete grafische Systeme erweist sich das Level-Konzept von GKS als unhandlich, da eine standardgerechte Implementierung des Levels 2c (und nur dieses Level besitzt die vollen Segment- und Eingabefähigkeiten) das grafische System mit einer Vielzahl von nicht unbedingt erforderlichen Funktionen aus den unteren Levels belastet. Weitere Probleme beim Einsatz des GKS ergeben sich durch

– das Speicherkonzept, das die Belange von dynamischen Bildänderungen und -manipulationen nicht ausreichend berücksichtigt
– das Segmentkonzept, bei dem abgeschlossene und existierende Segmente nicht verändert werden können
– die unzureichenden Mittel für eine dynamische Mehrfenstertechnik und eine parallele Visualisierung von Bildstrukturen.

Intensive Arbeiten bei der Weiterentwicklung der grafischen Programmiersysteme führten 1988 zur Standardisierung des in den USA entwickelten Systems PHIGS (Programmer's Hierarchical Interactive Graphics) /2/. PHIGS ermöglicht 2D- und 3D-Grafik, Bildmodifikationen auf elementarer Ebene und ein hohes Maß an Interaktion; es unterstützt Echtzeitänderungen. Neben dem Grundkonzept, das von GKS übernommen wurde, beinhaltet PHIGS folgende wesentliche Erweiterungen:

– hierarchischer Aufbau von Bildstrukturen
– dynamisches Attributkonzept
– Editierbarkeit von Bildstrukturen
– 3D-Grafik.

Im Bild 4 wird der hierarchische Aufbau für die grafischen Datenstrukturen in PHIGS aufgezeigt. Die Bildstruktur Y erbt die Umgebung, die bei ihrem Aufruf angetroffen wird. Von ihr selbst vorgenommene Änderungen gelten lokal und für alle durch sie aufgerufenen Strukturen. Die Verarbeitung von grafischen Attributen (Farbe, Linienstärke usw.) unterscheidet sich gleichfalls wesentlich vom Attributkonzept in GKS. Attribute werden in PHIGS in Abhängigkeit von dynamisch änderbaren Parametern ausgewertet und können zur Abarbeitungszeit der Bildstrukturen als Strukturelemente an jeder Stelle innerhalb der Struktur geändert werden. Durch dieses Attributkonzept in Verbindung mit der Strukturhierarchie bietet sich die Anwendung eines PHIGS-Systems für die Echtzeitanimation von Prozeß- und Steuerungsinformationen an. Der zur Implementierung von PHIGS erforderliche rechentechnische Aufwand (Rechenleistung, Speicherkapazität) steht jedoch gegenwärtig in Steuerungen für den prozeßnahen Bereich nicht zur Verfügung. Bekannte PHIGS-Systeme nutzen 32-Bit-Workstations mit einer UNIX-Umgebung /3/.

Die Verbindung zu den grafischen Ein- und Ausgabegeräten erfolgt über die *Geräteschnittstelle*. Neben der Anwendung von firmenspezifischen Geräteschnittstellen, wie GEM-VDI von Digital Research (1986) und

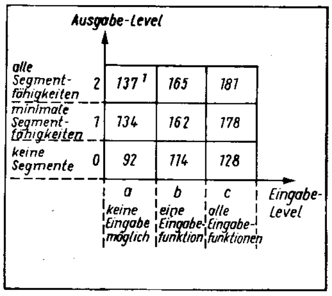


Bild 3 Funktionsumfang des GKS
1 – Anzahl der Funktionen

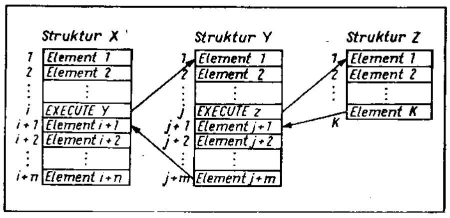


Bild 4 Strukturhierarchie für grafische Datenstrukturen in PHIGS

GDI von Microsoft (1986) für die Implementierung weit verbreiteter grafischer PC-Benutzeroberflächen (GEM, MS-Windows), richten sich gegenwärtig die Bemühungen der Hersteller grafischer Geräte verstärkt auf die Nutzung des ISO-Standards CGI (Computer Graphics Interface) /4/. CGI beinhaltet zur Zeit etwa die Funktionalität von GKS Level 2c. Tafel 1 verdeutlicht den Funktionsumfang dieser Schnittstelle. Mit CGI wird das Prinzip eines virtuellen grafischen Geräteinterfaces nach Bild 5 verwirklicht. Unterschiedliche Geräte können über eine gerätespezifische CGI-Codebindung in die jeweils gewählte CGI-Sprachbindung einbezogen werden. Nach Einschätzungen von Computergrafik-Spezialisten wird sich CGI zu einer wichtigen grafischen Schnittstelle entwickeln. Dieser Sachverhalt, verbunden mit solchen Eigenschaften wie

Tafel 1 Funktionsumfang von CGI

Funktionsgruppe	Anzahl der Funktionen
Steuerfunktionen, Fehlerbehandlung	29
elementare Ausgaben, Attributfunktionen	79
Segmentbehandlung	20
Eingabefunktionen	31
rastergrafische Funktionen	13

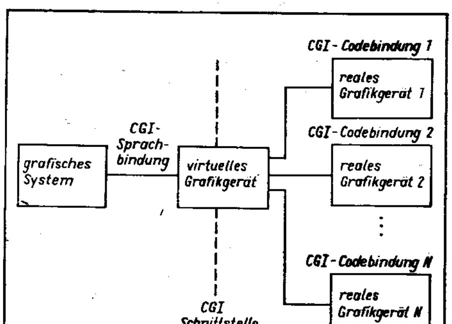


Bild 5 Prinzip einer virtuellen grafischen Geräteschnittstelle

– schnelles Modifizieren von Pixelmatrizen über effektive Rasteroperationen
– Kompatibilität zur Archivierungsschnittstelle CGM
– Berücksichtigung von CGI bei der Entwicklung neuer, hochintegrierter Grafikprozessoren,

favorisiert CGI als grafische Geräteschnittstelle auch für Anwendungen in Mikroprozessorsteuerungen.

Die *Archivierungsschnittstelle* dient dem externen Speichern von grafischen Datenstrukturen in sogenannten Bilddateien. Von besonderer Bedeutung auf dem Gebiet der grafischen Datenverarbeitung sind die beiden Bilddatei-Standards GKSM (Graphical Kernel System Metafile) und CGM (Computer Graphics Metafile).

GKSM ermöglicht den Austausch von Bildbeschreibungen zwischen verschiedenen GKS-Anwendungen. Im Prinzip handelt es sich bei einer GKSM-Datei um das Protokoll einer interaktiven Sitzung an einem GKS-System. Alle Aktionen während einer Sitzung werden in diese Bilddatei sequentiell abgespeichert. Für einen schnellen Zugriff auf externe grafische Datenstrukturen ist GKSM nicht geeignet.

Für schnelle grafische Systeme kommt zunehmend die Schnittstelle CGM zur Anwendung. Im Basisstandard /5/ sind sowohl die Funktionalität als auch die Sprach- und Codebindung festgelegt. Bild 6 zeigt den prinzipiellen Aufbau einer CGM-Datei. Die Kodierung kann binär, im ASCII-Code oder im Klartext erfolgen. Der Leistungsumfang des Basisstandards entspricht gegenwärtig etwa dem Level 0a im GKS. Ein wesentlicher Vorteil von CGM (gegenüber GKSM) besteht unter anderem darin, daß die einzelnen Bilder völlig unabhängig voneinander sind. Für CGM sind Erweiterungen um Segmentfunktionen und dynamische Bilderzeugung (CGM Add. 1) sowie 3D-Grafik (CGM Add. 2) bereits vorgesehen. CGM besitzt eine starke Funktionsorientierung zur Geräteschnittstelle CGI, so daß sich die gemeinsame Anwendung beider Schnittstellen innerhalb eines grafischen Systems anbietet.

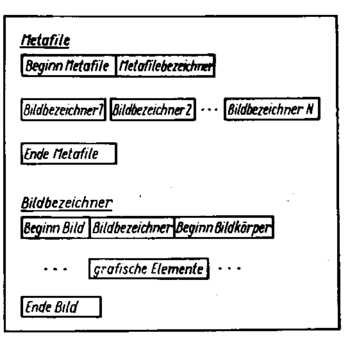


Bild 6 Struktur einer CGM-Bilddatei

Eingebettete Grafiksysteme

Die hohe Innovationsrate auf dem Gebiet der Prozessortechnik ermöglicht die Schaffung leistungsstarker grafischer Subsysteme für Anwendungen in 16-Bit- und 32-Bit-Mikroprozessorsteuerungen. Die Hardwarebasis dazu bilden Grafikprozessoren der 3. Generation, die seit etwa 1987 kostengünstig auf dem internationalen Markt verfügbar sind. Die Tafel 2 verdeutlicht die Leistungsfähigkeit dieser Grafikprozessoren. Auch wenn sich die innere Struktur dieser Prozessoren wesentlich voneinander unterscheidet, lassen sich doch die folgenden Merkmale als charakteristisch und leistungsbestimmend für diese neue Generation von Grafikschaltkreisen herausarbeiten:
– Integration von einer oder mehreren CPUs sowie Controllerbaugruppen auf einem Chip
– Hardware- bzw. Firmwareunterstützung für schnelle Grafikfunktionen (z. B. Rasteroperationen)
– eigenständige Bearbeitung komplexer Grafikfunktionen oder ganzer Grafikprogramme
– einfaches Host-Interface.

Für 16-/32-Bit-Industriebussysteme wie VMEbus, Multibus I und II werden Grafikbaugruppen mit diesen Prozessoren als Basishardware angeboten. Dabei reichen die Grafikfähigkeiten einer Konfiguration im Einfach-Europakartenformat von Zeichendarstellungen in Mehrfenstertechnik bis hin zur 2D-Echtzeitanimation von Steuerungskontrolldaten /6/.

Die Bilder 7 und 8 zeigen die typische Beschaltung der Grafikprozessoren i82786 und TMS34010 /7/, /8/. Der Systemspeicher in Bild 7 kann Grafik-Kommandolisten und andere Datenstrukturen beinhalten, auf die die CPU häufig zugreifen muß. Im Unterschied zum i82786, der einen technologisch festgelegten Satz von Grafikkommandos versteht, enthält der TMS34010 eine komplette 32-Bit-CPU mit einem entsprechenden Befehlssatz. Im dazugehörigen Systemspeicher (s. Bild 8) sind deshalb neben häufig genutzten Grafikkommandofolgen vor allem die speziellen Maschinenprogramme (Firmware), die den Umfang des mit der jeweiligen Baugruppe möglichen Grafikkommandosatzes festlegen, abgelegt.

Tafel 2 Charakteristische Parameter neuer Grafikprozessoren

Prozessortyp	HD63484	i82786	TMS34010
Hersteller	Hitachi	Intel	Texas Instruments
maximale Auflösung (Bildpunkte)	4096 × 4096	8192 × 4096	32768 × 32768
Anzahl der Farben	65536	256	65536
Bildschirmfenster	1 (Hardware)	beliebig (Software)	beliebig (Software)
Zeichengeschwindigkeit (Bildpunkte/s)	ca. 3 000 000		

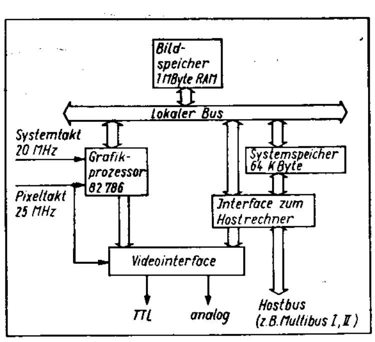


Bild 7 Grafikbaugruppe mit dem i82786

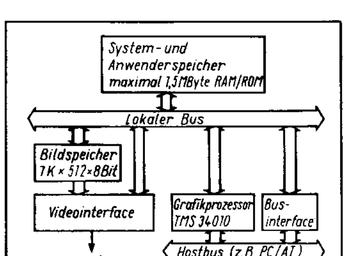


Bild 8 Grafikbaugruppe mit dem TMS34010

Die neuen Grafikprozessoren enthalten bereits auf der Prozessorebene eine Reihe von komplexen Grafikfunktionen, die üblicherweise erst in einer höheren grafischen Softwareschicht zur Verfügung stehen. Für einen optimalen Einsatz dieser Prozessoren in Mikroprozessorsteuerungen ist deshalb die funktionelle Schnittstelle für den Anwender aus dem konventionellen Bereich herauszulösen, und neue, komplexe Funktionen werden innerhalb der Geräteschnittstelle geschaffen. Obwohl bereits Bestrebungen erkennbar sind, auch komplexe grafische Schnittstellen wie GKS und CGI auf der Grafikbaugruppe selbst unterzubringen /9/, nutzen grafische Systeme für die Steuerungstechnik im prozeßnahen Bereich in der Regel einen auf die spezifischen Belange zugeschnittenen und damit reduzierten Funktionensatz /7/, /10/. Aufgrund der „harten" Randbedingungen erscheint ein solches Vorgehen auch für den nächsten Zeitraum die besten Lösungen für eingebettete grafische Systeme in Echtzeitsteuerungen zu ermöglichen.

Schlußbemerkungen

Aufbauend auf dem gegenwärtigen Kenntnisstand zum Einsatz grafischer Systeme in Mikroprozessorsteuerungen erfolgt am Institut für Automatisierung der AdW der DDR die Erarbeitung eines grafischen Basismoduls zur Steuerungsvisualisierung. Der Basismodul ist Bestandteil eines Rahmensystems für die farbgrafische Mensch-Maschine-Kommunikation in leistungsfähigen Mikroprozessorsteuerungen für den prozeßnahen Bereich. Die Grafiksoftware wird mit folgender Orientierung entwickelt:
– Anwendung hierarchischer Bildstrukturen, die einen Mechanismus zur Vererbung von Eigenschaften aufweisen sowie eine dynamische Belegung von Variablen während der Bildabarbeitungszeit ermöglichen
– Sicherung der Geräteunabhängigkeit
– Einsatz eines modernen Grafikprozessors und Einbettung des grafischen Basismoduls in eine Echtzeitumgebung.

Literatur

/1/ Encarnacao, J. L. u. a.: Graphische Datenverarbeitung mit GKS. München, Wien: Carl Hanser Verlag 1987
/2/ ANSI/X3H31: PHIGS Baseline Document. Document ANSI/X3H3/85–21, (X3H31/85–05), Revised 2/18/1985
/3/ PLOT 10 PHIGS. Prospekt der Fa. Tektronix, 1988
/4/ Arnold, D. B.; Bono, P. R.: CGM and CGI. Berlin, Heidelberg: Springer-Verlag 1988
/5/ International Standardization Organisation (ISO): Graphics-Metafile for Transfer and Storage of Picture Description Information (CGM). ISO DIS 8632 (1985)
/6/ Sommerkamp, A.; Ray, N.: VMEbus-Grafik für rauhe Industrieumgebung. mini Micro magazin, Heidelberg 4 (1988) 1–2, S. 106
/7/ Mayr, A.; Wlotzka, I.: Schnelle Grafik mit dem 82786. Elektronik, München 37 (1988) 11, S. 102
/8/ The MATROX PG – 1281. Prospekt der Fa. RAUSCHER GmbH, BRD, 1988
/9/ Asal, M. u. a.: The Texas Instruments 34010 Graphics System Processor. IEEE Computer Graphics and Applications, Oct. 1986, S. 24
/10/ Röhrle, J. u. a.: Neue CNC-Funktionen durch zugeschnittenes Grafiksystem. Wissenschaftlich-technische Zeitschrift industrielle Fertigung, 75 (1985) 6, S. 359

✉ KONTAKT ☎

Akademie der Wissenschaften der DDR, Institut für Automatisierung, Bereich Steuerungssysteme, Rudower Chaussee 5, Berlin, 1199; Tel. 6 74 36 52

Handelspartner stellen sich vor

Unter dieser Überschrift wollen wir westlichen Firmen die Möglichkeit geben, potentielle DDR-Kunden oder -Partner über das Leistungsangebot im Computersektor zu informieren.

Das wird für so manchen von Ihnen von Interesse sein – schließlich schaffen die neuen Bedingungen in der DDR nicht nur bessere Möglichkeiten, sondern auch Notwendigkeiten, sich über ausländische Handelspartner zu informieren. Dazu wollen wir zumindest in der Anfangsphase der neuen deutsch-deutschen Beziehungen unseren Beitrag leisten. Einen Anfang haben wir mit unserer Kontaktbörse im Märzheft bereits gemacht, nach den Firmenporträts wird die Zukunft dann aufgrund der Vielzahl der Handelspartner sicherlich bei kommerziellen Anzeigen liegen müssen.

Red. MP

SOFT-TRONIK

Ein Computerhandelshaus, seine Produkte und Leistungen

Das Unternehmen wurde 1985 von den beiden Inhabern und Geschäftsführern gegründet; in den letzten Jahren ist die Firma kontinuierlich gewachsen, 1989 wurde ein zweistelliger Millionenumsatz erzielt. Unsere Märkte liegen im In- und Ausland, vor allem auch in den RGW-Ländern.
Als Hersteller produzieren und vertreiben wir den Soft-Tronik-PC, unser eigenes Produkt. Die Komponenten werden aus Fernost importiert und hier zusammengebaut.
Als Distributor, d. h. als Großhändler, der das alleinige Vertriebsrecht für Händler und Endkunden in einem definierten Gebiet hat, vertreten wir die japanischen Hersteller Seikosha und Mitsubishi. Seikosha ist in der DDR nicht unbekannt, die Drucker findet man in den Intershopläden. Mitsubishi ist ein Hersteller von Computerperipherie, Monitoren und anderen Komponenten und gehört zu den weltweit größten Lieferanten. Darüber hinaus vertreiben wir Soft- und Hardwareprodukte aller namhaften internationalen Hersteller, wobei wir erhebliche Umsätze mit Netzwerkprodukten machen. Wir beschäftigen heute 30 Mitarbeiter – mit steigender Tendenz. Diese sind tätig im Bereich Technischer Service, zu dem der Kundendienst und das Lager gehören, hier arbeiten etwa 13 Mitarbeiter. In der Systemunterstützung und der Software sind 6 Personen beschäftigt, dann haben wir den Vertrieb mit 6 Mitarbeitern und schließlich die Administration und die Buchhaltung.
Die Mitarbeiter unserer Auslandsniederlassungen haben wir hier nicht gerechnet. Wir erbringen also alle die Leistungen rund um den Computer, die unsere Kunden von uns erwarten. Das beginnt bei der Beratung und hört bei der Schulung und der Betreuung der eingesetzten Systeme auf.

Der DDR-Markt – Risiken und Chancen

Natürlich wissen wir, daß auch auf dem Gebiet der Computertechnologie in der DDR ein gewaltiger Nachholebedarf besteht. Als Distributor von Seikosha und Mitsubishi werden wir auch die DDR betreuen, d. h. Servicestützpunkte und Niederlassungen errichten. Wir haben dies schon mit Erfolg u. a. in der Republik Polen getan und sind auch in Moskau und Leningrad mit Serviceniederlassungen vertreten. Wir suchen die Zusammenarbeit mit kompetenten DDR-Partnern, führen auch schon erfolgversprechende Verhandlungen, sind aber an weiteren Kontakten interessiert. Das soll recht schnell, also in allernächster Zukunft, geschehen.
Zur Zeit, d. h. im Februar, haben wir die devisen- und zollrechtlichen Regelungen des Interzonenhandels, an die wir uns zu halten haben, ebenso wie die Einschränkungen durch die COCOM-Liste. Aber, wir alle wissen es, so wird es nicht bleiben. Wir denken, daß wir unsere Kunden in der DDR bald ohne größere Probleme bedienen können, sonst hätten wir uns sicher nicht für den Eintritt in den DDR-Markt entschieden.

Überlegungen zu Vertrieb und Marketing

Wir entwickeln gegenwärtig ein Vertriebskonzept mit unseren Lieferanten; wichtig ist, daß die örtlichen Partner ganz eng eingebunden und natürlich stark unterstützt werden. Unsere Kunden werden wir vorwiegend bei kleinen und mittleren Betrieben suchen, besonders auch bei den neu entstehenden privaten Gewerbebetrieben und Freiberuflern. Einige Produkte sind sicher ebenso für große Betriebe interessant, man muß sehen, wie sich die wirtschaftliche Infrastruktur in der DDR entwickelt. Unser Konzept sieht jedenfalls eine intensive Bearbeitung des Marktes vor, was bedeutet, daß wir auch zu gegebener Zeit eine eigene Firma gründen wollen, an der sich z. B. Partner aus der DDR beteiligen können.

Einige Soft-Tronik-Produkte vorgestellt

Wir haben vor etwa 2 Jahren entschieden, nicht fertige PCs zu importieren, sogenannte No-names; einmal, um einen durchgehend hohen Qualitätsstandard sicherzustellen, zum anderen, um gefährliche Abhängigkeiten zu vermeiden. Die Qualitäts-Komponenten beziehen wir von leistungsfähigen Herstellern, z. B. Mainboards von Sertek, Laufwerke (Floppies und Festplatten) u. a. von Mitsubishi und NEC, Monitore von ADI und Mitsubishi, Cherry-Tastaturen, um nur einige zu nennen, und assemblieren diese Teile dann mit den entsprechenden Gehäusen zu den bestellten XTs oder ATs. Entscheidend ist, daß wir die Qualitätssicherung selbst in der Hand haben und durch umfangreiche Burn-in-Tests sicherstellen, daß der Kunde ein gutes Produkt erhält. Zur Zeit sind gerade die ersten 486er im Test. Unsere Preisliste umfaßt etwa 20 Standardkonfigurationen, die wiederum individuell den Kundenwünschen angepaßt werden können. Dazu kommen dann die verschiedenen Monitore und Drucker, je nach Anwendung.

Beratung des Kunden

Anhand der Problembeschreibung und des Mengengerüsts, wir gehen jetzt mal von Betrieben aus, erarbeiten unsere Mitarbeiter einen Vorschlag mit eigenem Angebot. Natürlich ist auch eine Demonstration der Hard- und Software möglich. Wichtig ist, daß der Kunde eine klare Problembeschreibung mit möglichst genauen Daten hat. Er sollte die Aufgaben präzise formulieren. Auch nach der Installation braucht der Kunde noch unsere Unterstützung. Diese erhält er über eine Schulung und die Einweisung in die Bedienung des Systems. Da gibt es, je nach Komplexität des gesamten Systems, verschiedene Möglichkeiten. Entscheidend ist: Der Kunde wird nach dem Kauf nicht allein gelassen.

Seikosha und Mitsubishi ein umfangreiches Angebot an Druckern und Peripherie

Seikosha ist, wie übrigens auch Epson, ein Unternehmen der Seiko-Gruppe, eines großen und traditionsreichen japanischen Herstellers. Die Produktpalette reicht vom preiswerten Matrixdrucker (9 oder 24 Nadeln) über den 18-Nadel-Hochgeschwindigkeitsdrucker mit 800 Zeichen/s bis zum Seitendrucker in LED-Technologie. Wichtig ist, daß Seikosha eine Fertigung in der Bundesrepublik aufgebaut hat, was natürlich große Vorteile für die Kunden bringt.
Die Geräte sind kaum störanfällig, man kann praktisch jedes Textproblem optimal mit einem der Seikosha-Modelle lösen. Wir haben auch Tintenstrahl- und Laserdrucker von Hewlett-Packard im Programm, um Kundenwünsche erfüllen zu können. Im Plotterbereich führen wir Systeme von Roland und Calcomp, beide kompetente Hersteller dieser Produkte.
Kommen wir zu Mitsubishi. Dieser Hersteller bietet ein umfangreiches Programm an Monitoren, von Standardmodellen bis zu hochauflösenden Colorgeräten mit einer Bildschirmgröße bis zu 37 Zoll. Diese Produkte werden häufig im CAD/CAM-Bereich eingesetzt. Im Scannerbereich ist der Handheldscanner interessant, der bis A4 mit hoher Auflösung und 16 Graustufen scannen kann. Bei den Druckern hat sich Mitsubishi auf Farbdrucker in Thermo-Transfertechnologie spezialisiert. Bis zu 4096 Farben können in bestechender Qualität wiedergegeben werden; Einsatzgebiete liegen im CAD/CAM-Bereich und in der Präsentationsgrafik. Neben den Ein-/Ausgabenmedien ist das „zweite Standbein" von Mitsubishi der Bereich der Speichermedien. Die Floppy- und Festplattenlaufwerke in 5,25 oder 3,5 Zoll gehören nach den branchenüblichen Benchmarktests zu den stabilsten und fehlerärmsten aller vergleichbaren Laufwerke. Lieferbar sind die Komponenten in verschiedenen Speicherkapazitäten. Wir verwenden diese Laufwerke übrigens in unseren eigenen PCs. Dann sind noch die sogenannten Optical Discs aufzuführen, die künftig mit ihren gigantischen Speichervolumen, z. Zt. etwa 600 Gigabyte, für besonders speicherintensive Anwendungen eingesetzt werden. Das ist ein grober Überblick über unsere heutige Angebotspalette, die natürlich je nach Kundenanforderungen erweitert werden kann. Wir sind jedenfalls darauf vorbereitet und zuversichtlich, gemeinsam mit unseren regionalen Partnern ein ebenso kompetenter und zuverlässiger Lieferant in der DDR zu werden.

soft-tronik
creative computer technology

Soft-Tronik GmbH

***Geschäftsführer:* Herr Lampka, Herr Osowiecki**
***Vertriebsleiter Inland:* Herr Burda**

Heidestr. 52
D–1000 Berlin 21
Tel. 394 40 03, 394 40 61
Telex: 17 308 203 SOFD, Telefax: 394 10 69

Mikado

Von der Software zum Systemhaus

Angefangen hat es nicht untypisch: Mehrere Kollegen einer großen Firma überlegen sich: Man müßte etwas Eigenes machen!

So wurde das Unternehmen 1983 von Spezialisten aus dem Hause Berthold, einem der bekanntesten Satzmaschinenhersteller, gegründet. Ursprünglich war man ein Softwarehaus, stellte aber bald fest, daß die Kunden nach einem Komplettangebot verlangten: Hard- und Software aus einer Hand.
Der Systemhausgedanke war damit geboren.

Die Gründer kamen aus der Welt der VAXen von DEC, entschlossen sich aber – nach einer Orientierungsphase – zur Konzentration auf die Welt der Mikros, der PCs.

Sehr bald diversifizierte das Ganze, die PC-Entwicklung explodierte förmlich, der Markt der Standardsoftware bildete sich, die PC-Peripherie nahm ungeahnte Formen an: Grafikkarten, Bildschirme, Laserdrucker, Tintenstrahler, Scanner galt es zu beherrschen.
Dies bedeutete auch für Mikado eine stürmische Entwicklung: Bis heute ist das Haus auf 21 Mitarbeiter angewachsen. Weiter stützt man sich auf externe Mitarbeiter für Programmierung und Werbung und hat seit einiger Zeit eine Tochter: Mikado-Saar, acht Mann hoch und auf CAD im Bauhauptgewerbe spezialisiert.

Der Systemhausgedanke – ein modernes Dienstleistungsverständnis

Als Systemhaus bietet Mikado in ausgewählten Bereichen und Branchen dem Kunden eine komplette EDV-Lösung. Im Unterschied zum gängigen Fachhandel versteht sich ein Systemhaus als ein Dienstleistungsunternehmen. Der Kunde erhält neben der entsprechenden Hard- und Software eine kompetente Beratung, das System wird nach seinen spezifischen Anforderungen konfiguriert und installiert. In diesem Sinne ist Mikado hauptsächlich auf Lösungen im gewerblichen und industriellen Bereich spezialisiert. Die Firma hält für diesen Kundenkreis nicht nur einen schnellen und zuverlässigen Hardwareservice bereit, sondern weist Mitarbeiter vor Ort in ihre neuen Systeme ein. Mikado unterhält einen eigenen Schulungssektor und bietet Kunden in einem After-Sales-service qualifizierte Betreuung auch nach Projektabschluß.
Um dieses umfangreiche Dienstleistungsprogramm zu garantieren, konzentiert sich Mikado auf die Produkte einiger weniger Hersteller, die eine Spitzenposition am Markt einnehmen und aufgrund einer hohen Produktqualität dem Kunden und Mikado als Händler gleichermaßen Zufriedenheit bescheren.

Partner Mikados sind Hewlett-Pakkard und Fujitsu. In einem neuen Vertriebskonzept unterstützt jetzt Hewlett-Packard ausdrücklich den Systemhausgedanken. Mikado trägt als einziges Sevice-Center Berlins das Prädikat „autorisiertes Systemhaus". Der Hardware-Lieferant Hewlett-Pakkard und das Systemhaus Mikado setzen mit diesem Konzept einen deutlichen Akzent. Im Bewußtsein, gerade im Unternehmens- oder öffentlichen Bereich Anlagen für hochkomplizierte Lösungsverbunde zu stellen, soll Dienstleistung gleichberechtigter Bestandteil in einem umfassenden Hardware-Vertrieb werden.

Neue Vertriebswege für Hardware-Lieferanten

HP-Systemhaus zu sein, bedeutet für Mikado, gemeinsam mit Hewlett-Pakkard an den Markt zu treten. So wird Mikado erstmalig den neuen HP VECTRA 486er PC mit dem EISA-Bus vorstellen, eine Power-Station mit 20 MIPS, die in den Bereich der mittleren Datentechnik und der Workstations einbricht. Besonders interessant wird der VECTRA für Anwendungen im CAD-Bereich, in PC-Mehrplatzsystemen, für DTP und umfangreiche Datenbankverwaltungen. Im PC-Bereich heißt das, neben dem Betriebssystemstandard MS-DOS wieder verstärkt mit dem Multi-user-System UNIX zu arbeiten. Die Entscheidung ist hier für das UNIX des Marktführers Santa Cruz Operation (SCO) gefallen.
Hewlett Packard und Mikado signalisieren durch ihre Zusammenarbeit ihren Kunden, daß hochkomplexe Technologien wie der VECTRA sich erst durch den angemessenen und kompetenten Support eines Dienstleisters rechnen.

Vom Systemverbund zur Individual-Software

Fünf Hauptbereiche von Mikado sind zu nennen:

Bürokommunikation
– Der Bereich Bürokommunikation, der ein komplettes Angebot mehrplatzfähiger Systeme bereithält, um alle im Büro- und Verwaltungsbereich anfallenden Arbeiten und Kommunikationsbedürfnisse zu bewältigen. (Mikado-Kontakt: Jan Holger Otto)

Kommerzielle Datenverarbeitung
– Der Bereich kommerzielle Datenverarbeitung, der mit der Software des Marktführers KHK die gesamte Palette vom Angebot über Auftragsabwicklung, Rechnungswesen, Lagerhaltung, Mahnwesen, Lohn- und Finanzbuchhaltung abdeckt. (Mikado-Kontakt: Barbara Heßler)

Grafische Datenverarbeitung
– Der Bereich grafische Datenverarbeitung, der um die Satz- und Layoutsoftware Ventura Publisher herum den ganzen Grafikbereich abdeckt mit Programmen zum Scannen, zur Bildbearbeitung, Geschäftsgrafik, Illustrationsgrafik, OCR, Schriftgrafik u. v. m.
Wenn auch etwas unscharf, so wird's doch häufig in dem Wort DTP zusammengefaßt. Das hauseigene DTP-System hat für viele Unterlagen Mikados schon gute Dienste geleistet, billiger und schneller als die Druckerei ist es allemal. (Mikado-Kontakt: Michael Mittelhaus)

Netzwerke
– Technisches Rückrat des Ganzen ist der Netzwerkbereich: Mikado ist in Berlin (und nicht nur dort) als Spezialist für NOVELL-Netze bekannt. Und weil man am liebsten das verkauft, was man selber verwendet, so ist im Hause ein Netzwerk mit ARCNet und Ethernet-Strängen mit Novell Netware 386 als Betriebssystem installiert. (Mikado-Kontakt: Wolfgang Dürr)

Reimund Reiter, Geschäftsführer bei Mikado

Softwareentwicklung
– Last, but not least, der Bereich Softwareentwicklung; hier entstehen zumeist Datenbankanwendungen: mehrplatzfähige Adressenverwaltung, ein Paket für Hausverwaltungen *DOMINO*, eins für Rechtsanwälte *Data Jura*, eines für Mitfahrerzentralen *Car Line* u. s. w. (Mikado-Kontakt: Chris Kasprowiak)

In diesem Bereich wurde auch das jüngste und bis dato prominenteste Mikado-Kind geboren: *Archimedes*, das neue Dokumentations- und Archivierungssystem für den Büroalltag. Von jedem Arbeitsplatz im Büro können künftig Archivunterlagen aufgerufen, angesehen, abgelegt, vervielfältigt oder weitergeleitet werden. Archimedes ist ein Archivierungssystem, das mit optischen Platten arbeitet und aufgrund seiner Effektivität und Schnelligkeit eine Alternative zum Mikrofilm darstellt.

Etwas kleiner, aber nicht minder interessant, *Printer Polyglott*, die multilinguale Textverarbeitung. Zusammen mit internationalen Sprachwissenschaftlern entwickelt, erlaubt es in mittlerweile 136 verschiedenen Sprachen der Welt am Bildschirm und auf den gängigsten Druckern mit den Originalzeichen der Landessprachen bzw. des Dialekts zu arbeiten. Vertreten sind: slawische Sprachen wie Bulgarisch, Russisch, Weißrussisch, Lettisch, Ukrainisch ebenso wie Georgisch. Griechisch wie Finnisch und Ungarisch, Arabisch (!) wie Altgriechisch, Koptisch wie Japanisch (!). Basierend auf der Top-Textverarbeitung Word Perfect ist es ein an Vielfalt, Qualität und Leistung konkurrenzloses Produkt.

Der Einzugsbereich von Mikado ist vornehmlich Berlin-West, aber es gibt einige Installationen im Bundesgebiet und seit einiger Zeit auch in der DDR. So zum Beispiel bei der Akademie der Wissenschaften, die jüngst bei den Tagen der DDR in Baden-Württemberg überzeugend die Leistungen der Akademie mit einer Broschüre darstellen konnte, die auf einem DTP-System von Mikado entstanden war.

Mitbestimmung bei Mikado

Eine Besonderheit dürfte das Gesellschaftsmodell von Mikado darstellen: Die GmbH ist vollständig im Besitz von derzeit 6 Mitarbeitern. Mitinhaber können nur Mitarbeiter werden, die eine Mindestzeit in der Firma tätig sind, niemand darf Majorität bekommen. Der Geschäftsführer ist einer Kontrolle der Belegschaft und eines von ihr gewählten Organs unterworfen.

Vielleicht ist das ein Vorbild für die Zukunftspläne: Derzeit wird viel mit DDR-Bürgern und -Firmen über mögliche Kooperationen gesprochen, es zeichnen sich sehr interessante Möglichkeiten ab.

mikado GmbH

Mikado

Systemhaus für Mikrocomputer
Wielandstr. 13
D-1000 Berlin 12
Tel. 3 24 40 68/3 24 50 52

Computer-Club

Zusatzsoftware für Bildungscomputer

Beim VEB Robotron Meßelektronik „Otto Schön" Dresden wurden für den Bildungscomputer neben der Systemsoftware folgende Dienst- und Hilfsprogramme entwickelt, die das Spektrum der auf dem A 5105 lauffähigen Standardsoftware ergänzen:

• BASIC

UMLK6313.RMC Druck der Umlaute aus RBASIC mit Drucker K 6313
SLOT-R-W.BAS Speicherbereich anzeigen, lesen und schreiben auf beliebigem Slot
TYPEX.BAS Dateianzeige
BACD3BIC.RMC Einbindung Basicode in RBASIC
PIC→SCR.BAS Konvertieren von Bilddateien vom Typ PIC von MS-DOS nach SCPX 5105
TB*.* Treibernachladeprogramm-system (Basic-Booter)
REV.RMC, RFVC.RMC RAM-Floppy für Video-RAM
RFM.RMC, RFMC.RMC RAM-Floppy für RAM-Erweiterungsmodul
DATBANK Datenbanksystem
PLOTV24 V.24-Plotteranschluß im Basic (HP-GL)
PLOTXY Anschluß Plotter XY 4140
KOMBICOP komfortables BICOP

• **NETZ**

BICLAN*.* Softwarepaket lokales Netz für mehrere BICs
PHYLANTR.BAS, PHYLANTR.RMC Dauertest lokales Netz mit Fehlerstatistik
STNR.BAS Anzeige der am DIL-Schalter eingestellten Adresse des lokalen Netzes
FLAG.BAS, JAB.BAS Testprogramme für lokales Netz (Service-Prüfvorschrift)

• SCP

INSTBIC, SCREEN, KEY.MAC, KEY.COM Einstellen von Systemparametern
ZGEIN.COM Manipulieren von Zeichensätzen für SCPX 5105
SCHMALAM.SZG, SCHMAL.SZG, STANDARD.SZG, STANDAM.SZG Zeichensatzdateien für SCPX 5105
TR*.* Treibernachladeprogramm-system
CEN1.COM Centronics-Ausgabe über I/0-Port
REV.COM, RFVC.COM Bildschirm-RAM-Floppy-Treiber
RFM.COM, RFMC.COM Modul RAM-Floppy-Treiber
LOE.COM, SYSDV.COM, SUBRET.COM, SW.COM Hilfsprogramme für SUBM-Kommandofiles
SLOTDUMP.MAC, SLOTDUMP.COM Anzeige des Speicherinhaltes beliebiger Bänke
VERSION.INC, HEXDU.INC, PRNAD.INC Include-Dateien für Assemblerprogrammierung
GRPASCAL Nutzung der Grafikroutinen des ROM-BIOS unter Pascal
TRANS1.COM Umsetzung von KC-Basic-Programmen in RBASIC

VEB Robotron-Meßelektronik „Otto Schön" Dresden, PSF 211, Abt. KVK, Dresden, 8012

Gerhard Gärtner

Kennengelernt:

Pascal für Kleincomputer

Michael Lennartz, Ludwigsfelde

Wenn die nun halbwegs erschwinglich gewordenen Kleincomputer nicht als teure Spielzeuge verkommen sollen, ist es notwendig, daß Standardsoftware und Programmiersprachen für verschiedene Bedürfnisse und Interessen zur Verfügung stehen. Ein Beitrag dazu ist die von einem Dresdner Kollektiv entwickelte, leistungsfähige Pascal-Implementierung für Kleincomputer. Seit der Veröffentlichung in /1/ ist das Interesse daran spürbar gewachsen.

Der Pascal-Compiler steht für alle Kleincomputer-Typen aus der DDR-Produktion zur Verfügung und ist 1986 durch Weiterentwicklung aus HiSoft-Pascal, eine Version für den Sinclair ZX-Spectrum aus dem Jahre 1983, entstanden. Die Programme sind somit quelltextkompatibel für KC 85/1/2/3/4 und KC 87. Die folgenden Aussagen beziehen sich auf die Version 1.5 für den KC 85/3. Das Programmsystem besteht aus Editor, Compiler und Laufzeitsystem und setzt eine 16-KByte-RAM-Erweiterung voraus. Die in Arbeit befindliche Version 2.1 wird auch ohne Zusatzmodul lauffähig sein. Die Frage des Vertriebs ist allerdings gegenwärtig noch nicht endgültig geklärt. Die wirtschaftlichen Änderungen in der DDR bieten jetzt aber auch den Softwareentwicklern den gerade in rechtlicher und finanzieller Hinsicht dringend benötigten größeren Handlungsspielraum.

Der Editor

Der integrierte Zeileneditor erinnert in seinem Funktionsumfang an Basic 80. Ein Full-Screen-Editor wäre zeitgemäßer, ist aber bei 40-Zeichen-Bildschirmen mit einer Reihe von Problemen verbunden. Trotzdem – EDAS hat ihn auch.

Bequem ist die eingebaute Suchfunktion für Zeichenketten. Des weiteren kann man Zeilen mit einem Kommando komplett verdoppeln. Die übrigen Funktionen entsprechen dem Standard: Löschen, Einfügen und Neunumerieren von Zeilen, Sichern und Laden des Quelltextes, Unterkommandos für die Bearbeitung einer Zeile wie die Korrektur, das Löschen bis zum Zeilenende, das Verwerfen der Korrekturen usw. Als komfortable Zugaben sind eine schnelle Scroll-Routine und eine Erweiterung des Zeichensatzes mit Kleinbuchstaben zu nennen.

Die Steuerung des Editors erfolgt unter anderem über Parameter, die für die Zeit des Editierens erhalten bleiben, was aber zu ungewollten Nebeneffekten führt. Nach den Parametern
D 120,140 (Löschen der Zeilen 120 bis 140)
muß man mit
I 200,10 (Eingabe ab Zeile 120 mit Zeilenabstand 10)
weiterarbeiten, um den Zeilenabstand von 10 wieder einzustellen. Die verkürzte Eingabe
I 120 (Eingabe ab Zeile 120)
behält den Zeilenabstand 140 aus der vorhergehenden Löschanweisung bei.

Der Compiler

Ähnlich wie in Turbo-Pascal kann das Kompilieren in Richtung Speicher oder in Richtung Kassette geschehen. Dabei können auch Programme erzeugt werden, die ohne Speichererweiterung arbeiten. Der Compiler stellt abschaltbare Kontrollfunktionen für die Überwachung der Feldindizes, des Über- bzw. Unterlaufs bei der Addition und Subtraktion ganzer Zahlen (Gleitkommaoperationen werden immer überwacht), des Stacks und des Programmabbruchs durch die Stop-Taste zur Verfügung. Das Listen des Programms während des Übersetzens läßt sich ebenso wie die Hardcopy-Funktion steuern. Bei Fehlern werden die fehlerhafte Zeile, die Position in der Zeile und eine Zahl als abgekürzte Fehlermeldung angezeigt. Das Übersetzen läßt sich manuell unterbrechen.
Ein Fehler kann eine Reihe von Folgefehlern erzeugen, die auch alle angezeigt werden. Eine vergessene *PROGRAM*-Anweisung zu Beginn des Quelltextes erzeugt so auch bei kleinen Programmen etwa ein halbes Dutzend Fehlermeldungen an verschiedenen Stellen.

Der Befehlsumfang

Dieser Compiler orientiert sich mehr am Standard-Pascal denn an Turbo-Pascal. So fehlt der Typ *String* mit der zugehörigen Zeichenkettenverarbeitung. Es steht ein umfangreicher Satz mathematischer Funktionen zur Verfügung, unter anderem COS(r), SIN(r), EXP(r), LN(r), TAN(r), ARCTAN(r). Für die Tastaturabfrage wurde INCH implementiert, für die Speicherung von Daten auf Kassette stehen die Prozeduren TOUT und TIN bereit. Dem Basic-Anwender werden die Routinen POKE, PEEK und USER bekannt vorkommen, an Turbo-Pascal erinnert die INLINE-Anweisung zum Einfügen von Maschinencode. Vermißt habe ich Anweisungen zur Ausgabe auf einen Drucker. So bleibt bei den CAOS-Rechnern dem Programmierer wieder nur der Weg, mittels des Steuerzeichens CHR(15) die Hardcopy-Funktion des Rechners ein- bzw. auszuschalten, oder über die INLINE- und USER-Anweisungen in Richtung Maschinencodeebene zu gehen. Auch eine ABSOLUTE-Option zur Zuweisung einer absoluten Adresse an eine Variable fehlt.

Als sehr angenehm empfinde ich die Erweiterung der Formatsteuerung in der Form XXXX:n:H zur Darstellung in hexadezimaler Schreibweise.

Die Dokumentation

Die Kurzdokumentation /2/ ist mit 8 Seiten etwas mager geraten, aber für Anwender, die bereits mit Pascal gearbeitet haben, ausreichend. Anfänger und Umsteiger aus Basic benötigen unbedingt ein Lehrbuch.
Einige Formulierungen in der Kurzdokumentation sind irreführend. In der Beschreibung des Editors fand ich zunächst keine Funktion zur Eingabe von Quelltext. Dafür muß man I wählen, aber die Erläuterung dazu lautet *Automatische Erzeugung von Zeilennummern*. Die Eingabe von Quelltext ist wie in Basic auch durch Zeilennummer, Leerzeichen oder Text möglich. Das geht aus der Dokumentation aber nicht hervor.

Ähnlich erging es mir mit der Kontrollfunktion **P** des Compilers. Sie ist auf Bildschirm eingestellt und bewirkt umschaltend die Ausgabe auf den Drucker. Das sich diese Umschaltung auf das vom Compiler erzeugte Protokoll bezieht und nicht auf den erzeugten Code (es gibt ja keine explizite Druckanweisung!), ist aus der Anleitung nicht zu entnehmen.

Auch entspricht INCH nicht im engeren Sinne dem INKEY$ aus BASIC, denn INCH wartet, bis eine Taste gedrückt wird (siehe hierzu /1/). INKEY$ liefert üblicherweise einen leeren String, wenn keine Taste gedrückt ist. Das kann bei INCH nicht funktionieren, denn es gibt keine Stringverarbeitung und damit keinen String der Länge Null. Und eine Variable des Typs CHAR hat immer die Länge 1. In den zukünftigen Versionen wird INCH bei nicht gedrückter Taste CHR(O) liefern.

Diese Pascal-Compiler-Implementierung ist eine Bereicherung des Softwareangebots für alle Kleincomputer der DDR-Produktion. Eine stärkere Orientierung am Quasi-Standard Turbo-Pascal wäre sicher günstiger gewesen, insbesondere was den Editor und die Zeichenkettenverarbeitung betrifft. Die hohe Portabilität der Quelltexte macht dieses System sicher für alle interessant.

Literatur

/1/ Gutzer, H.: **Entenflug mit Tempo – der** PASCAL-Compiler machts möglich. Funkamateur, Berlin 38 (1989) 8, S. 379
/2/ Burmeister u. a.: PASCAL für Kleincomputer. VEB Robotron-Vertrieb Berlin, 1987.

Undokumentiert:

Das DOS-Compatibility-Flag

Weil die Kirschen in Nachbars Garten am süßesten schmecken, zieht es den Programmierer auch zu undokumentierten, aber nützlichen Funktionen. MS-DOS bietet eine Menge davon. Inzwischen Allgemeingut und zum Schreiben von Hintergrund-Programmen erforderlich ist das INDOS-Flag (Unterfunktion 34H im Sammelinterrupt 21H). Damit läßt sich umgehen, daß DOS nicht reentrant ist.
Dagegen wenig bekannt ist das Kompatibilitätsflag von DOS. Ab Adresse 40:0000H befindet sich der BIOS-Variablen-Bereich, und auf Adresse 41:5053H finden wir das DOS-Compatibility-Flag. Es repräsentiert eine Binärzahl zwischen 0 und 100 und gibt die Hardwarekompatibilität in Prozent an. Die restlichen Bits sind undefiniert und müssen ausgeblendet werden. Eventuell sind sie für folgende Versionen reserviert. Beim Booten des Systems wird das Flag aus der aktuellen Konfiguration errechnet. Grundlage dafür sind die Schalter auf der Hauptplatine bzw. die Angaben im CMOS-RAM.
Wie kann der Programmierer nun dieses Wissen nutzen? Häufig steht die

```
; Programm zum Anzeigen des DOS-Compatibility-Flags
;
; Uebersetzung: MASM comp;
;               LINK comp;
;               EXE2BIN comp comp.com

code     segment
         assume cs:code, ds:code, es:code, ss:code
         org    100h

start:   lea    dx,flag              ; Adresse des DOS-Comp.-Flags
         mov    cx,2                 ; 2 Ziffern (0 bis 99)
         lea    si,maske             ; Maskenadresse
         mov    ah,49h
         call   maske                ; ausblenden Bits > 100
         mov    ah,09                ; DOS-Fkt. 09 (Zeichenausgabe)
ausgabe: int    21h
         loopnz ausgabe              ; bis alle Ziffern ausgegeben

; Programm beenden

         mov    ah,4ch               ; DOS-Fkt. 4c (Prog. beenden)
         mov    dl,0
         int    21h

; Datenbereich, Adresse des DOS-Comp.-Flags

flag:    db     41h,50h,53h          ; Adresse 41:5053

; Unterprogramm zum Filtern der Bits fuer den Bereich 0 bis 99

maske:   dec    word ptr [si-1]      ; low byte minus 1, weil Bereich
                                     ; 0-99 statt 1-100
         and    [si],ah              ; ausblenden Bits > 100
         ret

code     ends
         end    start
```

Bild 1 Programm zur Ausgabe des DOS-Compatibility-Flags

Frage, ob es notwendig ist, (die Kompatibilität sichernden) DOS-Funktionen zu verwenden, oder ob aus Zeitgründen nicht lieber auf BIOS-Routinen oder direkt auf entsprechende Adressen (z. B. Bildschirmspeicher) zugegriffen werden soll. Dies kann jetzt mit einer Abfrage im Programm entschieden werden. Bild 1 zeigt ein Programm, das das Kompatibilitätsflag Ihres Rechners anzeigt.

Literatur

Bergler, T.: Betriebssystem-Forscher: Undokumentierte MS-DOS-Funktionen aufspüren. c't (1987) 9

Uwe Schulze

Formulardruck mit Nadeldruckern

Bernd Matzke, Delitzsch

Bei Anwendungen der Rechentechnik in der Praxis ist es oft notwendig, Daten in Form von Formularen auszudrucken. Da diese aus verschiedenen Gründen nicht immer durch einen reinen Computerausdruck ersetzt werden können, wird das Beschreiben vorgedruckter Formulare mit dem Drucker notwendig. Dabei ergeben sich Probleme, da das Format und die Anordnung der Formularfelder in der Regel nicht der Schriftgröße und dem Standard-Zeilenabstand der verwendeten Drucker entsprechen.

In den folgenden Prozeduren für die freie Positionierung des Druckkopfes eines zu Epson kompatiblen Druckers werden nur Druck-Funktionen verwendet, über die auch der LX 86 verfügt. Diese können auch auf den moderneren Druckern (LX 800, FX 1000) genutzt werden. Die freie horizontale Positionierung des Druckkopfes mit hoher Auflösung (1/60 Zoll) ist beim LX 86 nicht direkt möglich. Der Druckkopf wird dazu an den Zeilenanfang und der Drucker in den Grafikmodus gesetzt. Durch Aussenden von n Grafikbytes mit dem Inhalt 0 wird der Druckkopf um 1/60 Zoll nach rechts bewegt, ohne das Druckbild zu stören.

Auf diese Weise kann der Druckkopf innerhalb einer Zeile in jeder beliebigen Reihenfolge auf jede beliebige Position gebracht werden.

Der Papiervorschub erfolgt mit einer Auflösung von 1/216 Zoll, ist aber nur in eine Richtung möglich. Das Formular muß also zeilenweise beschrieben werden.

Weiterhin ist es günstig, die Koordinaten stets absolut zu einem Bezugspunkt anzugeben, damit sich kleinere Toleranzen nicht zur ungünstigen Seite hin addieren.

Als Bezugspunkt wurde die linke Kante des ersten Zeichens der ersten Druckzeile gewählt. Koordinatenangaben erfolgen in mm, die X-Werte werden von links nach rechts angegeben, die Y-Werte entsprechend der Richtung des Papiervorschubes von oben nach unten. Bei der Festlegung der Feldkoordinaten im Programm bzw. später beim Einspannen des Formulars ist zu berücksichtigen, daß der Abstand zwischen der 1. Druckzeile und der zur Positionierung benutzten Druckkopfoberkante nicht bei allen Druckern identisch ist.

Der Druck eines Formulars verläuft in drei Schritten:

① Grundstellung des Programms durch Aufruf der Prozedur NEUES_FORMULAR (Formularlänge). Dieser Prozedur wird die Formularlänge in Millimetern übergeben, damit später beim Formularauswurf kein unnötig langer Seitenvorschub erfolgt.

② Ausfüllen des Formulars durch SET_FORMULAR (Y-Koordinate, X-Koordinate, Schriftart, Text). Da durch die Prozedur die einzelnen Einträge zeilen- und spaltenweise sortiert werden, ist bei der Übergabe keine Rücksichtnahme auf die Reihenfolge der Einträge erforderlich.

Die Angabe der Schriftart gestattet die Auswahl aus mehreren Schriftbreiten und -arten (*Master-Select*), so daß bestimmte Einträge hervorgehoben werden können bzw. ein Einpassen in die Formularfelder ermöglicht wird.

③ Ausdruck mit PRINT_FORMULAR

Bei dieser Prozedur wird das Formular gedruckt. Dazu wird der Papierendefühler abgeschaltet, um bis an den unteren Rand drucken zu können. Der im Heap belegte Speicherplatz wird zurückgegeben.

Die angegebenen Prozeduren SET_ZEILE und SET_SPALTE können für alle Epson-Drucker verwendet werden. Verfügt der verwendete Drucker über den Befehl zur absoluten Positionierung des Druckkopfes **ESC $ n1 n2** (z. B. der FX 1000), so kann die Prozedur SET_SPALTE durch SET_SPALTE2 ersetzt werden, wodurch der Druckvorgang beschleunigt wird, ältere Modelle (LX 86) aber nicht mehr korrekt angesteuert werden.

```
  1: PROGRAM formular;
  2: CONST formfeed          = #12;         {Steuerzeichen fuer Epson-Drucker}
  3:       cr                = #13;
  4:       vorschub_set      = #27#51;
  5:       blattlaenge_set   = #27#67#0;
  6:       spalte_set        = #27#36;                      {nur fuer FX1000}
  7:       drucker_reset     = #27#64;
  8:       papierfuehler_aus = #27#56;
  9:       grafik            = #27#42#0;           {Grafik, einfache Dichte}
 10:       normal            = #27#33#0;
 11:       fett              = #27#33#8;
 12:       schmal            = #27#33#4;
 13:       breit             = #27#33#40;
 14:       zoll              = 25.4;
 15:
 16: TYPE s             = STRING[80];
 17:      pointer       = ^rec;
 18:      rec           = RECORD
 19:                        spalte,zeile:INTEGER;
 20:                        schriftart,
 21:                        text:s;
 22:                        next:pointer;
 23:                      END;
 24:
 25: VAR merkzeile:INTEGER;
 26:     anfang:pointer;
 27:     blattlaenge:INTEGER;
 28:
 29: PROCEDURE set_zeile(zeile:INTEGER);
 30: VAR n:INTEGER;
 31: BEGIN
 32:   n:=Round(216.0*(zeile-merkzeile)/zoll);     {umrechnung in 1/216 zoll}
 33:   WHILE n > 0 DO BEGIN          {es gehen maximal 255/216 Zoll auf einmal}
 34:     IF n > 255 THEN WriteLn(LST,vorschub_set,Chr(255))
 35:                ELSE WriteLn(LST,vorschub_set,Chr(n));
 36:     n:=n-255; END;
 37:   merkzeile:=zeile;                        {merken der eingestellten zeile}
 38: END;
 39:
 40: PROCEDURE set_spalte(spalte:INTEGER);
 41: VAR i,n:INTEGER;
 42: BEGIN
 43:   Write(LST,cr);                              {druckkopf an zeilenanfang}
 44:   n:=Round(60.0 * spalte  / zoll);             {umrechnen in 60stel zoll}
 45:   Write(LST,grafik,Chr(n MOD 256),Chr(n DIV 256));
 46:   FOR i:=1 TO n DO Write(LST,#0);            {n pixelspalten nach rechts}
 47: END;
 48:
 49: PROCEDURE set_spalte2(spalte:INTEGER);
 50: {nur fuer spezielle Drucker, z.B. FX 1000}
 51: VAR n:INTEGER;
 52: BEGIN
 53:   n:=Round(60.0 * spalte  / zoll);             {umrechnen in 60stel zoll}
 54:   Write(LST,spalte_set,Chr(n MOD 256),Chr(n DIV 256));
 55: END;
 56: PROCEDURE neues_formular(laenge:INTEGER);
 57: {Grundstellung + Einstellung Blattlaenge}
 58: BEGIN
 59:   merkzeile:=0;
 60:   New(anfang);
 61:   anfang^.zeile:=-1; anfang^.next:=NIL;
 62:   blattlaenge:=Trunc(1.0*laenge / zoll) + 1;
 63: END;
 64:
 65: PROCEDURE set_formular(y,x:INTEGER;schriftart,text:s);
 66: {Aufbau des Formulars, Sortieren nach Zeile und Spalte}
 67: VAR z,p:pointer;
 68: BEGIN
 69:   z:=anfang;
 70:   New(p);                                               {neuer Eintrag}
 71:   p^.spalte :=x;               p^.zeile:=y;
 72:   p^.schriftart:=schriftart; p^.text :=text;          {belegen Eintrag}
 73:   WHILE  (z^.next <> NIL)     AND           { nicht letzter Eintrag und }
 74:          ((z^.next^.zeile  < y) OR          { Zeile < neue Zeile    oder}
 75:           (z^.next^.zeile = y)   AND        {    Zeile = neue Zeile und }
 76:           (z^.next^.spalte < x))            {    Spalte < neue Spalte)  }
 77:     DO z:=z^.next;
 78:   p^.next:=z^.next;                               {Einfuegen neuer Eintrag}
 79:   z^.next:=p;
 80: END;
 81:
 82: PROCEDURE print_formular;
 83: VAR memo,eintrag:pointer;
 84: BEGIN
 85:   Write(LST,drucker_reset);
 86:   Write(LST,blattlaenge_set,Chr(blattlaenge));
 87:   Write(LST,papierfuehler_aus);
 88:   eintrag:=anfang;
 89:   WHILE eintrag^.next <> NIL DO BEGIN
 90:     memo:=eintrag;
 91:     eintrag:=eintrag^.next;
 92:     Dispose(memo);
 93:     WITH eintrag^ DO BEGIN
 94:       set_zeile(zeile);
 95:       set_spalte(spalte);
 96:       Write(LST,schriftart,text);
 97:     END;
 98:   END;
 99:   Dispose(eintrag);
100:   Write(LST,formfeed); Write(LST,drucker_reset);
101: END;
102:
103: BEGIN   {program formular}
104:   neues_formular(100);
105:   set_formular( 0,  0,normal,'links oben');
106:   set_formular(90,150,fett  ,'rechts unten');
107:   set_formular( 0,150,schmal,'rechts oben');
108:   set_formular(30,130,breit ,'rechts');
109:   set_formular(90,  0,schmal,'links unten');
110:   set_formular(60, 20,fett  .'links');
111:   set_formular(90, 60,normal,'unten Mitte');
112:   set_formular(45, 70,breit ,'Zentrum');
113:   print_formular;
114: END.
```

Bild 1 Prozeduren für den Druck von Formularen

Neue Software für die Koppelkarte SIE 2

Für den Anschluß von beliebigen seriellen Geräten an das Gerätesystem P8100 (speziell Echtzeitteil MMS 16) gab es bisher Anpassungsschwierigkeiten, da ein Uminitialisieren der SIE2 vom Masterprozessor (U 8000) nicht möglich war. Die eigene Lösung ersetzt die Firmware der SIE2. Wesentliche Möglichkeiten sind:

- Einstellbarkeit aller SIO-Parameter für jeden Kanal
- Neuinitialisierung der CTC
- Jedem Kanal können bis 512 Byte übergeben werden, die dann selbständig gesendet werden.
- Die empfangenen Daten werden bis 512 Byte/Kanal gepuffert.
- Das Abholen der Daten erfolgt vorzugsweise durch eine Interruptbehandlung.
- Jeder Kanal kann beim Senden unterbrochen werden.
- Alle Kanäle sind gleichberechtigt.

Eine umfangreiche Beschreibung mit Programmierbeispielen wird (auf Diskette) mitgeliefert.

VEB Nachrichtenelektronik Greifswald, Abteilung H/HE, Brandteichstraße 25, Greifswald, 2200; Tel. 6 62 93 *Holz*

Modula-2 für den P8000

An der Humboldt-Universität, Sektion Mathematik, wurde der Modula-2-Compiler **m8** für den P8000 entwikkelt. Er übersetzt in vier Pässen Modula-Quelltext in Assemblerquelltext des C-Assemblers **cas**. Auf Assembler- und Objektkodeebene können die aus Modula generierten Files mit C- und Assemblerfiles zu einem ausführbaren und segmentierten Programm verbunden werden. Zur Arbeitsumgebung gehören das Programm **m8make**, welches aus den Modula-Quellen ein Eingabefile für das Unix-Programm **make** erzeugt, und ein **Full-screen-Editor**, der im Fehlerfall aufgerufen wird und die Positionierung auf fehlerhafte Zeilen gestattet.

Humbold-Universität zu Berlin, Sektion Mathematik, Unter den Linden 6, Berlin, 1086; Tel. 20 93 27 21 *Dr. Grafik/Werner*

comFORTH-CrossCompiler Version 5.x

Die als CrossCompiler bezeichnete Komponente der comFORTH-Systemfamilie gestattet die Erzeugung zugeschnittener Forth-Kernsysteme für die verschiedensten Anwendungsfälle. So ist sowohl die Generierung großer Entwicklungssysteme für PCs als auch kleinster ROM-fähiger Applikationen für die Prozessorfamilien Z8, Z80, 8086 und TMS320 möglich. Die Version 5.x weist folgende Vorteile auf: Die Übersetzergeschwindigkeit wurde auf ein Mehrfaches erhöht. Der Wortschatz des Systems ist in allen Betriebsarten Forth-83-kompatibel. Die Segmentierungskriterien wurden in bezug auf Konsistenz und Speichereffizienz überarbeitet. Die Möglichkeiten zur Compilererweiterung mit Macros wurden vergrößert. Die Bedienoberfläche wurde weitestgehend an die des verteilten Compilers comFORTH-plus angepaßt. Die Portabilität des CrossCompilers wurde verbessert.

Albert-Einstein-Straße 2, Rostock, 2500; Tel. 40 52 02 *Dr. Pfüller*

EC 1834 als Hostcomputer von CAD-Arbeitsplätzen

Zur Eignung des EC 1834 als Hostcomputer von CAD-Arbeitsplätzen wurden umfangreiche Untersuchungen durchgeführt. Besonderer Wert wurde auf die Erprobung der sogenannten ESER-PC-Mischvariante gelegt, die zwei Direktsteckverbinder auf der Grundplatine aufweist. Dieses Gerät erlaubt das Testen eines umfangreichen Sortiments von Import-Komponenten (Grafikkarten, Grafikbildschirme, Digitizer, Plotter). Dieses Sortiment beinhaltet auch jüngste Erzeugnisse mit höchstintegrierten Bausteinen, z. B. Grafikkarten der ARTIST- und ATLAS-Familie, sowie diverse EGA-Varianten. Softwarebasis der Untersuchungen waren international verbreitete Grafiksysteme, wie CADdy, Autocad und die bauspezifische CAD-Software der Firma Nemetschek (BRD).
Der Untersuchungsbericht enthält Details, die den fachlich vorbelasteten Nutzer befähigen, Inbetriebnahmen importierter Hardwarekomponenten und die dementsprechende Softwareparametrierung selbst durchzuführen. Wir bieten den Bericht zur entgeltlichen Nachnutzung an.

Bauakademie der DDR, Institut für Projektierung und Standardisierung, Abteilung CAD-Technik, Plauener Straße 163–166, Berlin, 1092; Tel. 37 83 23 42 *Lenz*

Units für Turbo-Pascal

Für die Programmiersprache Turbo-Pascal wurden speziell für A 7150, EC 1834 und Schneider-PC Units entwickelt, um u. a. Inkompatibilitäten der Unit Dos zu korrigieren. Ebenso stehen Units für die Bildschirmgestaltung zur Verfügung.
Es werden Units für folgende Problemstellungen als dokumentierter Quelltext angeboten: Fenstertechnik, Pull-down-Menüs, Eingabemasken, Filebearbeitung, Blättern und Suchen im Textfile, Mikroeditor (WS-kompatibel, 255 Zeilen), Kursorverwaltung, Plotterschnittstelle (A 7150–K 6418), Druckerschnittstelle (alle Druckertypen). Die Units für Bildschirmgestaltung können sich farbmäßig auf den entsprechenden Monitortyp einstellen.

Technische Hochschule Ilmenau, Sektion Gerätetechnik, WB Technische Optik, Ilmenau, 6300 *Beyer*

V.24-Hardware-Interrupt

Für die Kopplung des A 7150 mit anderen Rechnern und peripheren Geräten wurde eine Softwarelösung entwickelt, die es gestattet, Daten interruptgesteuert zu übernehmen. Die Lösung beinhaltet die entsprechende Initialisierung der seriellen Schnittstelle V.24 auf der ASP sowie die Änderung der Interruptmaske des PIC, so daß der Interrupt des SIO an die CPU weitergeleitet wird. Durch den Interruptmodus wird festgelegt, ob der Interrupt bei jedem Zeichen oder nur beim ersten Zeichen (bei blockweiser Übertragung) ausgelöst werden soll. Die Interruptroutine, die im Anwenderprogramm eingebunden ist, liest den SIO aus und legt die Daten auf einen Puffer im RAM. Die Software liegt als Unit vor und kann in eigene Turbo-Pascal-Anwendungen eingebunden werden.

VEB Schwermaschinenbaukombinat „Ernst Thälmann" Magdeburg, Abt. MTZ, Marienstraße 20, Magdeburg, 3011; Tel. 68 27 57 *Dr. Krüger*

Datenübertragung von Nicht-ASCII-Dateien

Zur Übertragung von Daten oder Dateien zwischen verschiedenen Rechnersystemen hat sich das ASCII-Format durchgesetzt. Dennoch steht der Anwender gelegentlich vor dem Problem, auch Dateien übertragen zu müssen, die Kodierungen enthalten, welche zu Transferproblemen führen. Zur Lösung dieses Problems wurde von uns folgender Weg beschritten: Das Programm **nobin** wandelt im Bedarfsfall die kritischen Dateien in eine übertragbare, dabei aber trotzdem möglichst kurze Form um. Danach erfolgt die gewünschte Datenübertragung, beispielsweise mit KERMIT, CCSMV, ISCMAN, REMOTE o. ä., und das Kommando **bin** führt anschließend die korrekte Rückwandlung durch.
Die Programme bin und nobin liegen z. Zt. für folgende Systeme vor: MUTOS 1700, WEGA (P8000), VENIX-86, XENIX-286, 2.9 BSD, CP/M 2.22, CP/M-86, MS-DOS. Die Programme wurden in C geschrieben, so daß praktisch keine Grenzen für weitere Portierungen existieren. Die Bedienung der Kommandos erfolgt in der unter UNIX üblichen Form der Filter einheitlich für alle Betriebssysteme. Die Weitergabe zur Nachnutzung erfolgt als Quelltext und in compilierter Form.

Bergakademie Freiberg, Organisations- und Rechenzentrum, Abt. Systemprogrammierung/Basissoftware, Bernhard-von-Cotta-Straße 1, Freiberg, 9200; Tel. 51 28 43 oder 51 20 42. *Weber*

Festplattenüberblick mit ALLDIR

Es wurde das Programm ALLDIR für das Betriebssystem MS-DOS erarbeitet, das einen vollständigen Überblick über alle Dateien einer Festplatte (oder Diskette) mit den Angaben Dateiname und -extent, Größe, Attribute, Modifikationsdatum und -uhrzeit sowie Pfad gibt. Die Suche beginnt und endet im eingestellten Pfad und schließt alle Subdirectories ein. Dabei werden alle Dateien oder nur ausgewählte ausgegeben. Bei fehlerhaftem Aufruf wird ein Hilfstext angezeigt. Die Ausgabe kann auf Bildschirm oder in eine Textdatei erfolgen. Diese Datei kann mit einem Datenbanksystem ausgewertet werden. Das Programm kann eingesetzt werden für: Bestandsaufnahme aller Dateien, Suchen bestimmter Dateien, Feststellen von Modifikationen des Dateibestandes (Registrieren von Aktivitäten), Vorhandensein der Dateien, Dateigrößen, Uhrzeit und Datum. In einer Anwendung zur Systemsicherung wird automatisch ein Differenzprotokoll gegenüber der letzten Sicherung erzeugt.

VEB Rohrtechnik Delitzsch, Friedrich-Ebert-Straße 30, PSF 61, Delitzsch, 7270; Tel. 36 36 *Dr. Wernicke*

Zwei Bildschirmarbeitsplätze an einem PC

In unserem Betrieb wurde eine Lösung für die Kopplung eines zweiten Bildschirmarbeitsplatzes (Monitor und Tastatur) mit einem Personalcomputer des Typs A 7100, A 7150 oder EC 1834 entwickelt. Der zweite Bildschirmarbeitsplatz kann räumlich getrennt (max. 20 m) vom Rechnergrundgerät aufgestellt werden. Da diese Rechnertypen kein Echtzeitbetriebssystem besitzen, kann jeweils nur ein Bildschirmarbeitsplatz arbeiten. Die entsprechende Auswahl erfolgt manuell durch eine Umschalteinheit am Rechnergrundgerät. Als vorteilhaft erweist sich ein PC mit Festplatte, wobei jedem Nutzer eine Partition zugewiesen wird. Damit entfällt der nachteilige Diskettenwechsel, und eine Fernsteuerung der Umschalteinheit wird möglich.
Hardwareseitig ist der Aufbau der Umschalteinheit und die Installation eines zweiten Monitors mit Tastatur erforderlich.

VEB Farbenglaswerk Weißwasser, Abteilung Rechentechnik, Straße der Befreiung 104, Weißwasser, 7580 *Lehmann/Rössel*

Entwicklungssystem für Einchipmikrorechner

Angeboten wird ein Entwicklungssystem für Einchipmikrorechner der Typen U 88xx, welches insbesondere für 8-Bit-Rechner der unteren Leistungsklasse konzipiert wurde. Die Entwicklung von Programmen erfolgt mit einem Objektkodeeditor, wobei auch Pseudobefehle und Kommentare verarbeitet werden können. Das Testen der Programme kann von der Hardware unabhängig durch einen Simulator erfolgen. Durch den Einsatz eines zum Lieferumfang gehörenden 4-KByte-PROM-Emulators kann das Programm ohne Programmieren eines PROMs im Echtzeitlauf auf dem zu entwickelnden EMR-System gestartet werden. Das Programm kann durch den Anwender erweitert werden, dazu wird dem Nutzer eine Vielzahl von Systemrufen zur Verfügung gestellt (u. a. Assembler, Reassembler). Der Programmkern ist an kein Betriebssystem gebunden, alle Consolen- und Gerätezugriffe werden über einen Sprungverteiler

geführt. Anpassungen existieren zur Zeit für den KC 85/1 und das Betriebssystem SCP.

Medizinische Akademie Erfurt, Institut für Pathologische Biochemie, Nordhäuser Straße 74, Erfurt, 5010, Tel. 79 23 59

Otto/Herzberg

Hintergrundausgabe von Plotfiles

Der DZT 90 × 120/RS ist ein sehr exakt, aber langsam arbeitendes Gerät. Während der Zeichenarbeit ist der Rechner für die Ausgabe von Daten an den DZT blockiert, obwohl für die Übertragung nur etwa 1 % der Zeichenzeit benötigt wird. Aus diesem Grund wurde ein Programm für den A 7100 entwickelt, welches nach dem Laden des Plotfiles und dem Initialisieren der V.24-Schnittstelle die Daten im Hintergrund an den DZT ausgibt. Der Rechner steht während der Zeichenarbeit dem Nutzer wieder zur Verfügung.
Bereits vorhandene DZT-Treiber müssen entsprechend modifiziert werden: Das dem Treiber zur Ausgabe übergebene Zeichen wird nicht an den DZT übertragen, sondern in ein Plotfile geschrieben. Bestehende Software, die den Treiber nutzt, kann ohne Änderung im Einsatz bleiben.

VEB Schwermaschinenbau „Karl Liebknecht" Magdeburg, Abt. WEV, Alt Salbke 6–10, Magdeburg, 3011; Tel. 4 32 28 20

Rhode

U 884-Entwicklungssystem unter DCP

Im IfAM Erfurt wurde ein U 884-Entwicklungssystem geschaffen, das auf einem Standard-16-Bit-PC (1834, 7150, XT, AT) läuft und mit einem Entwicklungsmodul (U 884-Emulator vom P8000 des VEB EAW oder einem kompatiblen Modul) über eine COM-Schnittstelle zusammenarbeitet. Ein Serienterminal ist nicht erforderlich, und eine Nutzung als reines Softwareentwicklungssystem (ohne Entwicklungsmodul) ist ebenfalls möglich. Die derzeitige Version ermöglicht nur die Softwareentwicklung für den internen 4-KByte-ROM-Bereich. Die Entwicklung erfolgte in Turbo-Pascal. Das Programm realisiert eine integrierte Entwicklungsumgebung mit den Komponenten WordStar-kompatibler Editor (mit Block- und Suchkommandos), Assembler (mit Listingerzeugung und Datenbereitstellung für das Debugging) und Debugsystem, die über Befehlstasten gesteuert wird. Analog zu Turbo-Pascal 5.0 wird beim Debugging das Listing auf den aktuellen PC-Stand positioniert, und gesetzte Breakpoints werden im Listing dargestellt. Nach Veränderung der Quelle läuft mit einem einzigen Tastenbefehl der gesamte Prozeß *Source-Save → Assemblierung → U 884-Reset → U 884-Memory-Load → Listing auf 000C positionieren* automatisch ab, so daß das Debugging unmittelbar danach fortgesetzt werden kann.

VEB Mikroelektronik „Karl Marx" Erfurt, Stammbetrieb, IfAM, Abt. CI3, PF 194, Rudolfstraße 47, Erfurt, 5010; Tel. 6 21 02/34

Dr. Conrad

GRAFADIA

GRAFADIA ist ein zeichenorientiertes Pseudografik-Programm für den KC 85/3, 4 zum Entwickeln, Ändern, Speichern sowie zum Drucken von Zeichnungen mit folgenden Eigenschaften: einfache Bedienung, störungssicher, Zeichnungsgröße A4 oder A3 (druckerabhängig), frei definierbarer, nachladbarer Grafikzeichensatz (integrierter Grafikeditor) für spezielle Darstellungen verschiedener Gewerke (128 Zeichen zu 8 × 8 Bildpunkten), beliebige Tastaturbelegung mit den Grafikzeichen, Fenstertechnik (12 frei definierbare Fenster in Bildschirmgröße), Baugruppen-(Symbol-)arbeit – Hardware: KC 85/3, KC 85/4 mit oder ohne Diskettenlaufwerk. Hauptanwendungsgebiete für das Grafiksystem sind Übersichtsgrundrisse, Schaltschemata und Projektzeichnungen in o. g. Formaten. Das Programm wird bei Zusendung von 1 Kassette oder Diskette zum Preis von 500,– Mark vertrieben.

Zwischengenossenschaftliche Einrichtung Landbauprojektierung Zwickau, Abt. RT, Reichenbacher Straße 158, Zwickau, 9502; Tel. 78 82

Fritzsch

MS-DOS-Routinen für Fortran-Programme

Zur Vervollkommnung des Fortran-Funktionsvorrates bis zu einem Niveau, das andere Programmiersprachen (z. B. Turbo-Pascal) schon seit langem bieten, und zur Erhöhung der äußeren Attraktivität von in Fortran geschriebenen Programmen wurden bei uns Unterprogramme entwickelt, die auf die Funktionen des Betriebssystems MS-DOS (ab Version 3.20) zugreifen. Sie sind prinzipiell auf allen verfügbaren Fortran-Compilern lauffähig. Im wesentlichen lassen sich damit folgende Funktionen realisieren: Veränderung von Bildschirmfarben und -attributen, Postionieren des Kursors und Löschen von Bildschirmfenstern, Zugriff auf Kommandozeilenparameter, auf Systemdatum und -zeit, Beeinflussung von Laufwerk und Pfad, Aufruf anderer Programme im aktuellen Pfad, Auflisten des Directories und daraus über Kursortasten Auswahl von Dateien sowie Menüaufwand. Die Unterprogramme liegen in Form einer Bibliothek vor, die im weiteren die systemspezifischen Maschinenprogramme enthält, so daß relativ problemlos weitere MS-DOS-Funktionen in Fortran-Programme eingebunden werden können. Zur Nachnutzung werden außer dieser Bibliothek die Fortran-Quelltexte sowie eine Kurzbeschreibung der Routinen angeboten.

Technische Universität Karl-Marx-Stadt, Sektion Automatisierungstechnik, WB 14220, Reichenhainer Straße 70, Karl-Marx-Stadt, 9022

Kleineberg

INTED/INTLINK/INTARC

Das Programm wurde in Turbo 5.0 geschrieben. Die Bedienung des Editors für Bildschirmmasken ist stark an die Steuerzeichen für Textverarbeitungssysteme angelehnt und damit leicht zu erlernen. Es existieren zwei Programmvarianten. *Variante 1* ist für Programmierer gedacht und unterstützt das Generieren von Bildschirmmasken, die in ein Pascal-Programm eingebunden werden können. Der gesamte Nutzerdialog wird damit einfacher und vor allem übersichtlicher gestaltet. Das Programm verfügt über eine Laufzeitbibliothek in Form von 4 Units, die dem Programmierer die verschiedensten Funktionen zur Verfügung stellen. Der Zugriff auf die gespeicherten Masken (bzw. Windows) ist durch eine parallel angeordnete Datenstruktur wesentlich schneller als der vieler inhaltlich vergleichbaren Programme. *Variante 2* richtet sich an reine Nutzer und bedarf keiner Programmiersprachenkenntnisse. Mit dem Maskeneditor werden Texte erfaßt und farbig oder hell/dunkel ausgestaltet. Schlüsselwörter müssen über eine Editor-Sonderfunktion markiert werden und sind dadurch mit einem weiteren Bild verbunden. Der Linkmodul erzeugt für die Datendatei ein .EXE-File, welches das Durchblättern der eingegebenen Fenster sowie einige Sonderfunktionen ermöglicht, so z. B. das Markieren von Seiten und das Anspringen derselben ohne mehrfaches Zurückblättern.

Für die Programme INTED (Editor), INTLINK (Linkmodul), INTARC (Archivierungsmodul) und die Laufzeitbibliothek existieren Dokumentationen; außerdem gibt es ein Demonstrationsprogramm für Variante 2.

Technische Hochschule Leipzig, Sektion Automatisierungsanlagen, Wissenschaftsbereich Automatische Steuerung, Karl-Liebknecht-Straße 132, Leipzig, 7030; Tel. 39 43/1 79

Tennert

Steuerung für Bearbeitungs- und Montagezellen

Eine sinnvolle Gliederung im Material- und Informationsfluß läßt sich ausgehend von einem Zellenkonzept erreichen. Eine Produktionszelle kann eine Fertigungszelle, eine Montagezelle, eine Meßzelle o. ä. sein.
Softwarefunktionen der von uns entwickelten Zellensteuerung CAMARS-ZEKO_2.0 sind: Koordination der Funktion mehrerer Bearbeitungs-, Montage- und/oder Transporteinheiten, DNC-Betrieb für CNC- und/oder IRS-gesteuerte koordinierte Einheiten, einschließlich Rüsten und Versorgung mit Werkstücken und Werkzeugdaten, Führung und Überwachung der CNC-, IRS- oder PC-gesteuerten Bearbeitungs-, Montage- oder Transportfunktionen koordinierter Einheiten, Editieren verwalteter und neuer NC-Programme, Protokollierung und automatisierter (Wieder-) Anlauf sowie Netzbetrieb für Einbindung in Mehrzellenstrukturen (PC-LAN)

Soft- und Hardware-Basis: C, DCP; ESER-PC, A 7150, ICA oder IBM-Kompatible. Nutzungswünsche für CAMARS-ZEKO_2.0 versehen Sie bitte mit einer Kurzcharakteristik des Einsatzfalls (Struktur der Zelle, Art der Einheiten, Steuerungshardware, Einführungsdatum).

VEB Mikromat Dresden, TRS, Mügelner Str. 36, Dresden, 8036

Huth

Grafikeditor GREDI

GREDI ist ein Grafikeditor für 16-Bit-Mikrorechner unter MS-DOS mit CGA-Grafikkarte und mit entsprechendem Bildschirm (Grafik: 320 × 200, 4 Farben).

Auf der Grundlage der Grafikfunktionen von Turbo-Pascal wurde ein Programm geschaffen, welches dem Anwender mit komfortablen Bedienfunktionen ein schnelles Editieren von Grafiken auf dem Bildschirm ermöglicht. GREDI ist ein Malprogramm für Computer ohne Maus. Der Kursor wird über die Kursortasten sowie die Tasten Home, End, Pgup, Pgdn gesteuert. Mit Hilfe einer mitgelieferten Prozedur (Turbo-Pascal im Quelltext) können diese Bilder vom Programmierer in jedes Anwendungsprogramm eingeblendet werden.
Das Programm unterstützt das Erzeugen und Anwenden von Symbolen in der Größe 20 × 20 Pixel. Diese Bilder sind in 9 Dateien mit je maximal 100 Symbolen gespeichert. Der Aufruf erfolgt automatisch. Bei der Programmanwendung werden alle Einzelschritte in einem Protokoll gespeichert. Der Preis beträgt 550 M.

Bau- und Montagekombinat Ost, Betrieb FPT, BT Schönefeld, am Flughafen, Schönefeld, 1189

Brand

Pascal für KC 87

Für Rechner vom Typ KC 85/1 und KC 87 wurde ein Pascal-Erweiterungsmodul entwickelt, der jetzt auch für den Bevölkerungsbedarf zur Verfügung steht. Das Pascal-System ist kompatibel zu Turbo-Pascal und beinhaltet alle Funktionen, die ohne Diskettenarbeit möglich sind. Durch den Modul wird es möglich, die auf den Heimcomputern erarbeiteten Programme ohne Änderung auf eine BC oder PC zu übernehmen. Der Modul besteht aus acht 4-KByte-EPROMs 2732 und den notwendigen Decoder- und Treibeschaltkreisen. Er belegt den Speicherbereich ab 4000H. Zum Lieferumfang gehören Modul und Dokumentation.

Labor für Elektronik-Leiterplatten, Fa. Haselbauer, Georg-Schumann-Straße 323, Leipzig, 7026; Tel. 59 10 26

Dr. Haselbauer

Wir suchen ...

... ein Gerätetreiberprogramm für den A 7100/EC 1834 zum Anschluß eines Seriendruckers SC 1152 P/01 an die IFSP-Schnittstelle.

VEB Projektierung Wasserwirtschaft, Betriebsteil Stralsund, Badenstraße 18, PSF 54/55, Stralsund, 2300; Tel. 69 60

von Wedelstädt

... jemanden, der uns unsere Interfacebox IF 3000 so programmiert, daß wir die Möglichkeiten der Erika 3004 besser ausnutzen können.

Torsten Schöne, Karl-Marx-Straße 34, Bitterfeld, 4400; Tel. 4 16 06

Entwicklungen und Tendenzen

Intels neuer Supercomputer

Intel Scientific Computers, ein Unternehmensbereich der Intel Corporation, stellte im Januar den neuen parallelen Supercomputer mit i860-Prozessoren vor. Das iPSC/860-System ist das erste einer völlig neuen Produktreihe. Es verfügt über maximal 7,6 GFLOPS numerischer Rechenleistung und soll im Vergleich mit einer Cray Y-MP um den Faktor zehn kostengünstiger sein. Das System ist von 8 Prozessorknoten, mit insgesamt 480 MFLOPS Rechenleistung, bis auf 128 Knoten ausbaubar und somit den entsprechenden Preis-/Leistungsbedürfnissen leicht anzupassen. Die Rechner sind mit Interfaces für TCP/IP-Ethernet, VMS-Link und NFS einfach in Rechnernetze zu integrieren und an Hostsysteme zu koppeln.
Entwicklungspartner von Intel in Europa sind CHORUS Systemes in Frankreich und Strand Software Technologies in Großbritannien. CHORUS Systemes implementiert das verteilte Betriebssystem CHORUS auf iPSC-Rechnern, ein Betriebssystem, das eine Echtzeit-Unix-Umgebung auf allen Knoten ermöglicht. Damit wird Echtzeit-Datenerfassung und Signalverarbeitung auf einem iPSC/860-System möglich. Strand Software Technologies portiert die parallele Sprache Strand 88 Mitte des Jahres auf das neue System.
Die Einsatzgebiete liegen im Finanzwesen, in der Luft- und Raumfahrtindustrie, in der Forschung allgemein und in den Marktsegmenten traditioneller Supercomputer.
In Europa wird das erste iPSC/860-System an die Gesellschaft für Strahlen- und Umweltforschung (GSF) in München/Neuherberg geliefert. Es soll dort hauptsächlich in den Bereichen Umwelt- und medizinische Strahlenforschung verwendet werden.
Die Technische Universität München wird das neue System als Computerserver in einem großen Sonderforschungsbereich einsetzen, die Arbeiten im TOPSYS (tools for Parallel Systems)-Projekt fortsetzen und neue Algorithmen untersuchen. *MP*

SPARCserver und 80486-Board von Sun

Die US-amerikanische Firma Sun Microsystems bietet seit Ende vorigen Jahres zwei neue Rechner auf der Basis ihres SPARC-Konzeptes an. SPARC bedeutet übersetzt skalierbare Prozessorarchitektur; sie ist aus der RISC-Architektur (Computer mit reduziertem Befehlssatz) hervorgegangen. Das SPARC-Konzept erlaubt unter anderem ein Prozessordesign, bei dem die Strukturen eines Prozessors maßstabsgetreu verkleinert werden können. Das soll in Zukunft eine bedeutende Geschwindigkeitserhöhung ermöglichen.
Besonders für Datenbankmanagement mit mehreren hundert Workstations ist der schrankgroße **SPARCserver 490** geeignet. Er arbeitet mit einem 64-Bit-Speicherbus und einem Hochleistungs-VMEbus, bietet einen Hauptspeicher von 160 MByte und eine Plattenkapazität von 132 GByte. Für die „Einstiegsversion" mit 32 MByte RAM und 2 GByte Plattenkapazität müssen stattliche 315 000 DM aufgeboten werden.
Wesentlich interessanter für die Bürovernetzung ist der **SPARCserver 1**, der einem flachen PC ähnelt. Mit einem maximalen Arbeitsspeicher von 16 MByte und bis zu 1,3 GByte Plattenkapazität leistet er 12,5 MIPS (Millionen Instruktionen pro Sekunde). Als Netze können das NFS von Sun oder auch heterogene PC-Netze dienen. Auch hier gibt es eine Einstiegsversion mit 8 MByte RAM und 654 MByte Plattenkapazität – für rund 20 000 DM.
Aber auch die Anwender konventioneller Sun-Technik mit 80386-Prozessoren, wie der Sun386i (siehe auch MP 10/88, Seite 316), können ihre Geräte in kürze mit einer 80486-Austauschkarte verbessern. Dadurch kann die Leistungsfähigkeit dieser Rechner (bei einer 25-MHz-Taktfrequenz) von 5 auf 12 MIPS erhöht werden – allerdings zum stolzen Preis von 11 200 DM. *MP*

Neues von Bull

Eine neue Minicomputerserie kündigte die Bull AG, Köln, unter der Bezeichnung Bull DPS 6000 an. Die Serie besteht aus den Modellfamilien 200, 400 und 600, die unter dem einheitlichen Betriebssystem GCOS 6 arbeiten. Gegenüber den bisherigen Spitzenmodellen des Minicomputer-Angebotes Bull DPS 6 PLUS bieten die neuen DPS 6000-Rechner die dreifache Leistung und eine verdoppelte Anschlußkapazität. So reicht das verfügbare Leistungsspektrum nunmehr von 0,7 bis 10 MIPS, und es können bis zu 1 000 Terminals angeschlossen werden. Es bestehen pro System zwischen einem und vier Zentralprozessoren sowie zusätzlich (ab Modell 400) ein unabhängiger, eigenständiger UNIX-Peer-Prozessor zur Verfügung.
Die Bull-Gruppe, Paris, hat Ende Dezember vergangenen Jahres die Computeraktivitäten der amerikanischen Zenith Electronics Corp., Glenview/Illinois, übernommen. Damit rückt Bull in der Rangliste der weltgrößten Informationstechnik-Anbieter nach eigenen Angaben auf Platz sieben vor. *MP*

Erster 486-Portable

Hersteller eines tragbaren PCs, der als erster den neuen Intel-Prozessor 80 486 verwendet, soll die US-Firma Dolch Computer Systems sein. Die Rechenleistung des V. P. A. C. 486/25 mit dem 25-MHz-80 486 wird mit 10,8 MIPS angegeben. Die Speicherkapazität beträgt 2 MByte RAM, erweiterbar bis zu 16 MByte, und 40 MByte auf der Festplatte. Zur Grundausstattung des Koffergerätes gehören ein 1,2-MByte-Diskettenlaufwerk sowie ein 16-Graustufen-Monitor. Bis zu vier Steckkarten voller Länge haben im Gehäuse Platz, weitere Karten in einem sogenannten Back Pack. *MP*

ISDN-Fernkopierer von Siemens

Die Deutsche Bundespost (Telecom) hat dem ersten ISDN-Fernkopierer eine allgemeine Zulassung erteilt.
Der Siemens HF 2405 ist ein Telefaxgerät der Gruppe 4 der CCITT-Norm, dessen Leistung und Funktionen durch ein von Siemens entwickeltes Kommunikationsteil auf die Vorteile von ISDN ausgerichtet sind. Eine A4-Seite (CCITT-Normbrief) wird in neun Sekunden fernkopiert; und dies in einer Qualität und Übertragungssicherheit, die herkömmliche Fernkopierer nicht erreichen können.
So steht mit bis zu 400 dpi realer Auflösung (16 Punkte/mm) bei Gruppe-4-Betrieb die doppelte Aufzeichnungsqualität im Vergleich zu Faxgeräten der Gruppe 3 zur Verfügung. Der HF 2405 ist abwärtskompatibel und kann auch auf Gruppe-3-Funktionen zurückschalten, wenn am anderen Ende der Leitung noch kein ISDN-Gerät vorhanden ist.
Ein Dokumentenspeicher (1 MByte) ermöglicht Funktionen wie zeitversetztes Rundsenden oder programmierten Abrufbetrieb. Das Gerät hat Kurzwahlspeicher für 99 Rufnummern und 32 Zielwahltasten. Außerdem lassen sich die Rufnummern nach Namen alphabetisch sortiert über Buchstabentasten aufrufen. Die Bedienerführung erfolgt am Display über Klartext. *MP*

Strukturiert programmieren mit PCI-Basic

Mit der Programmiersprache PCI-Basic stellt Siemens ein Software-Paket vor, das alle Vorteile von Basic in sich vereinen soll. Sie ist einfach und leicht erlernbar, bietet großen Komfort beim Erstellen von Programmen und ist dialogorientiert aufgebaut.
PCI-Basic verfügt über Merkmale, die bisher nur bei Pascal und der systemnahen Sprache C bekannt waren. Solche Merkmale sind vor allem für eine modulare strukturierte Programmierung hilfreich, wie Module, Unterprogramme und vielfältige Kontrollfluß-Anweisungen, die ein strukturiertes Programmieren ermöglichen. PCI-Basic ist als Compiler-Basic aufgebaut. Die Programmiersprache arbeitet mit einer hohen Rechnergenauigkeit und -geschwindigkeit (Arithmetikprozessor-Unterstützung) sowie voll unterstützter umfangreicher Matrizenrechnung und Matrix-Ein-/Ausgabe.
PCI-Basic bietet einen Programm-/Datenbereich bis 640 KByte, verfügt über integrierte vielfarbige Grafiken in Fenstertechnik und Viewports, eine Skalierung in physikalischen Einheiten sowie zweidimensionales Umwandeln von Bildern, wie Drehen, Verschieben und Vergrößern. Ferner hilft PCI-Basis den Programmierer bei der IEC-Bus-Programmierung. Dazu werden besondere Befehle nach den in der neuen Norm IEEE 488.2 festgelegten Ausführungen angeboten. *MP*

Neue Tektronix-Workstation

Mit einer Rechenleistung von 17 MIPS sieht sich die Firma Tektronix als Anbieter der schnellsten Auftisch-Workstation der Welt. Dennoch ist die nach dem RISC-Prinzip arbeitende Tek XD 88/10 als neues Einstiegsmodell gedacht – zum Preis von etwa 45 000 DM. Ausgestattet ist die Maschine mit 8 MByte RAM, ausbaufähig bis zu 32 MByte, und 156 MByte Festplattenkapazität, erweiterbar bis zu 3 GByte. Zum Lieferumfang gehört ein Farbmonitor mit 1280 × 1024 Punkten und serienmäßig 256 Farben.
Haupteinsatzgebiete sollen CAD/CAM, Produktionsautomatisierung, Prozeßsteuerung und andere technische Anwendungen sein; dem entspricht auch der vorhandene VME-bus-Anschluß. *MP*

Multiprozessorsystem von Mitac

Nach Compaqs Systempro (siehe MP 3/90, 4. Umschlagseite) präsentierte jetzt auch der taiwanesische Hersteller Mitac ein Multiprozessorsystem. Der **Mitac 500** kann mit bis zu sieben 80386-Prozessoren (25 MHz, ohne Wartezyklen) bestückt werden und besitzt zwei Bussysteme; als Prozessorbus den C-Bus und für die E/A-Einheiten den AT-Bus. Es können bis zu 160 Benutzerterminals gleichzeitig am System arbeiten, die Leistung des voll ausgebauten Rechners liegt bei bis zu 25 MIPS. Als Betriebssystem wird das SCO Unix **MPX** verwendet. *MP*

576 Neuronen auf einem Wafer

Eine Neuentwicklung auf dem Gebiet der neuronalen Netze gab die japanische Firma Hitachi bekannt. In ihrem Zentrallabor gelang es, ein neuronales Netz mit 576 Neuronen (48 Chips zu je 12 Neuronen) auf einem 5-Zoll-Siliziumwafer zu integrieren. Die Neuronen wurden in einer 0,8-μm-Technologie hergestellt.
Das Problem bei dieser Entwicklung bestand in der hohen Zahl der zur Verknüpfung notwendigen Synapsen. Sie ist im allgemeinen etwa um den Faktor 100 höher als die Zahl der Neuronen. Den Entwicklern gelang es, Synapsen und Neuronen gleicher Anzahl zu verwenden und damit den Aufwand zu reduzieren. Dementsprechend hoch soll auch die Arbeitsleistung sein. Ein spezifisches Optimierungsproblem soll mit dem neuen neuronalen Netz in 60 ms lösbar sein, während ein Supercomputer dafür gegenwärtig 6 Stunden benötige.

Quelle: Elektronik. – München 38 (1989) 22. – S. 6 Fa

Lichtleiter auf Siliziumbasis

Die Entwicklung eines neuen Grundmaterials für Plastik-Lichtleiter, das für Wellenlängen zwischen 500 und 2000 nm transparent sein soll, gelang den Bell Laboratorien (USA). In diesem Material sind nicht nur organische Kunststoff-Anteile vorhanden, sondern zusätzlich auch Silizium. Der neue Werkstoff trägt die Bezeichnung Polysylene.
In weiteren Forschungsarbeiten will man die „Pulsübertragung in optischen Wellenleitern" untersuchen.
Obwohl von den Bell Laboratorien noch keine detaillierten Angaben über das Material veröffentlicht wurden, geht man davon aus, daß sich mit diesem Material nicht nur Fasern, sondern auch optische Leiterstrukturen auf Siliziumsubstraten herstellen lassen. Die dafür notwendige UV-Maskentechnik wäre bei den Bell Laboratorien vorhanden.

Quelle: Elektronik. – München 38 (1989) 22. – S. 6 Fa

Farbechte Darstellung von 16 Fenstern auf Farbbildschirmen

Farbbildschirme werden sich bei Personalcomputern immer stärker durchsetzen. Betrug ihr Anteil im Jahre 1986 erst 50 Prozent, soll er 1990 80 Prozent und 1992 sogar 90 Prozent erreichen. Die Auflösung von 640 × 480 Bildpunkten (VGA) ist bereits weit verbreitet.
Mit dem Einsatz der Fenstertechnik ergeben sich bei den Farbbildschirmen Probleme der farbechten Wiedergabe der einzelnen Bildschirmausschnitte. Das zuletzt dargestellte Fenster bestimmt die Farbe, die den bereits auf dem Bildschirm vorhandenen Darstellungen aufgedruckt wird. Dadurch werden diese Farben verfälscht.
Für die Ansteuerung von Farbbildröhren hat sich das Prinzip der Farbpaletten-ICs durchgesetzt, Bausteine, die drei RAMs und drei D/A-Umsetzer enthalten. Um eine richtige Wiedergabe der Farben in allen Fenstern zu erreichen, müßte bei der Darstellung eines neuen Fensters die Farbpalette mit neuen Daten geladen werden. Das ist zeitlich nicht vertretbar. Eine andere Lösung wären zusätzliche Farb-ICs, die bei Eröffnung eines neuen Fensters hinzugeschaltet werden. Der Aufwand würde sich dann bedeutend erhöhen.
Die Firma Brooktree, Kalifornien, hat einen Baustein entwickelt, der 4 Farbpaletten-RAMs und die Steuerelektronik enthält. Dadurch sollen die Kosten gegenüber den Einzelchips reduziert werden können. Mit dem Baustein lassen sich 16 Fenster auf dem Bildschirm gleichzeitig farbecht darstellen.

Quelle: Elektronik. – München 38 (1989) 22. – S. 32, 34 Wie

32-Bit-PC in Buchgröße

Die Firma Matsushita (Japan) kündigte einen auf der Basis des 80386SX entwickelten neuen Personalcomputer an, der sich durch seine minimale Größe auszeichnet. Nach eigenen Angaben soll das Gerät in jede „Umhängetasche" passen.
Um diese geringe Größe zu erreichen, wurden 0,25 mm starke Keramikplättchen als Träger für die Chips verwendet und diese in sechs Lagen übereinander gestapelt. Damit konnte das Volumen um 80 Prozent verringert werden. Das Gerät soll weniger als 3 kg wiegen.
Die Firma will im Sommer 1990 mit der Serienproduktion beginnen. Die dafür notwendige Produktionsstätte wurde mit einem Kostenaufwand von rund 225 Mio DM in Kobe errichtet.

Quelle: Elektronik. – München 38 (1989) 22. – S. 6 Fa

EL-Display mit Speichereffekt

Im Gegensatz zu bisherigen Elektrolumineszenz-Displays bietet das von dem französischen National d' Etudes des Telekommunications (CNET) entwickelte EL-Flachdisplay eine integrierte Speicherfunktion. Dazu wurde in Dünnschichttechnik auf die EL-Schicht eine Fotoleiterschicht (PC-Schicht) gelegt. Diese beiden Schichten erzeugen zusammen einen Effekt, der als „bistabiles optisches Verhalten" bezeichnet wird. Die Ansteuerung erfolgt wie bei „normalen" EL-Displays. Das am einzelnen Bildpunkt emittierte Licht erzeugt in der PC-Schicht lokal begrenzte Zonen, in denen der spezifische Widerstand auf einem niedrigen Wert gehalten wird. An diesen Punkten bleibt die Lichtemission erhalten, auch nach Abschalten des Aktivierungsimpulses.
Gegenwärtig ist nur eine Schwarzweißdarstellung möglich. Mit dieser Technik soll aber der Leistungsbedarf um den Faktor 10 geringer sein, während gleichzeitig die Helligkeit um den Faktor 2 verbessert wurde. Diese Angaben beziehen sich auf Laptops mit EL-Displays.
Die gegenwärtig erreichte Auflösung beträgt 640 × 400 Bildpunkte.

Quelle: Elektronik. – München 38 (1989) 23. – S. 6 Fa

Betriebssysteme und PCs

Während der Einsatz von Betriebssystemen mit Multitasking- und Multiuser-Fähigkeiten stark expandieren wird, sollen leistungsfähige PCs als Stand-alone-Rechner kaum noch zur Anwendung kommen.
Eine Analyse der Verkaufszahlen von PCs im Jahre 1988 ergab, daß die meisten PCs in den USA abgesetzt werden konnten. Es folgten Großbritannien und die BRD. Für 1989 prognostizierte das US-amerikanische Marktforschungsunternehmen IDC, daß sich die Rangfolge zwischen Großbritannien und der BRD umkehren wird. Dabei bezieht sich das Marktforschungsunternehmen nur auf die Klasse der Bürorechner (ohne Homecomputer).
Bei den Betriebssystemen ist das Einplatzsystem MS-DOS das führende. Das Mehrbenutzersystem Unix hatte 1988 nur einen Anteil von 1,2 %, der sich aber 1990 auf 5 % weltweit erhöhen soll.

Quelle: Elektronik. – München 38 (1989) 21. – S. 18 Fa

Aufwände für die Mikroelektronik-Produktion

Ein Laborarbeitsplatz der Mikroelektronik kostet 250 000 DM. Bis zum Jahre 2000 wird sich ein derartiger Arbeitsplatz auf 500 000 DM verteuern. Der Aufbau von Fertigungsstätten wird bis dahin Investitionen in Höhe von 14 Mrd. DM erfordern.

Um eine dem Verbrauch adäquate Position in der Mikroelektronik aufzubauen, werden in der BRD F/E-Mittel in Höhe von 21 Mrd. DM bis zum Jahre 2000 nötig. Die Mitarbeiterzahlen in F/E-Bereichen müßten von rund 2000 auf 6500 anwachsen, ohne die Kapazitäten der Hochschulen.

Die Kosten einer Produktionslinie für 1-μm-Technik liegen bei 300 Mio DM, für eine Submikrometer-Linie bei 500 Mio DM. Mit den kleiner werdenden Strukturen kann eine Produktionslinie im Jahre 2000 schon 1 Mrd. DM kosten.

Quelle: Elektronik. – München 38 (1989) 24. – S. 49, 50 Wie

US-Verteidigungsministerium erhöht Ausgaben für Rechentechnik

Die Marktforschungsfirma Frost & Sullivan geht davon aus, daß auch bei einer drastischen Reduzierung des Budgets des US-Verteidigungsministeriums die Ausgaben für moderne Computertechnik erhöht werden. Außerhalb des aus einem Sonderkonto finanzierten „rechnerintensiven" SDI-Programms sollen bis 1995 etwa 327 Mio Dollar für Computertechnologie ausgegeben werden, was mehr als eine Verdoppelung der Ausgaben bedeuten würde.
Geplant ist die Entwicklung von Parallelrechnern, die die Leistung derzeitiger Computer um das Hundert- bis Tausendfache übertreffen. Diese sollen in „intelligente" Waffen eingesetzt werden, die Entscheidungen zum Angriff selbständig treffen. Damit würden zum SDI-System kongruente Waffen entstehen.
Schwerpunkte der Entwicklung sind neben den Hochleistungsrechnern: hochentwickelte Softwaretechnologien, die optische Informationsverarbeitung und neurale Netze als erweiterte Möglichkeiten der Simultanverarbeitung.
Diese Aktivitäten sind im sogenannten FCCSET-Programm (Federal Coordination Council for Science, Engineering and Technology) zusammengefaßt, das nach gegenwärtigen Schätzungen etwa ein Viertel der Forschungsmittel umfaßt, die bis jetzt für Hard- und Softwaretechnologien geplant sind.

Quelle: Die Computer Zeitung. – Leinfelden-Echterdingen 20 (1989) 26. – S. 4 Wie

Vakuumdrucken halbiert Leiterplattenkosten

Ein neues Verfahren zur Herstellung von Leiterplatten ist das Vakuumdrucken elektrisch leitender Druckfarben. Es wurde von der amerikanischen Gesellschaft Printon Inc. und deren Tochterunternehmen Atek Systems Inc. entwickelt. Nach deren Ansicht würden viele Fertigungsstufen, wie die Maskenherstellung, das Fotografieren, das Ätzen, das Waschen, das Verzinnen und das Lotbeschichten, entfallen. Außerdem würde sich die Zeit, die bisher notwendig war, um eine Leiterplatte herzustellen, von eineinhalb Stunden auf anderthalb Minuten verringern. Mit dem neuen Verfahren sollen sich nach Ansicht der Entwickler auch die Herstellungskosten der fertigen Leiterplatte halbieren lassen. Damit eignet es sich besonders für kleine Losgrößen. Das Verfahren kann auch für Leiterplatten jeglicher Art angewendet werden. Es liefert Leiterbahnen von 0,08 Millimeter mit 0,05 Millimeter Teilung.
Das neue Hochgeschwindigkeits-Vakuumdruck-Verfahren benutzt ein geschütztes Druckverfahren mit metallographischer Druckfarbe sowie ein Bestrahlungssystem zum Aushärten. Ein Druckkopf in der Plattenverarbeitungseinheit läßt dabei ein Drucksieb im Vakuum auf das Plattensubstrat fallen. Durch das Vakuum wird eine genau bemessene Menge Druckfarbe durch das Sieb angesaugt und gleichmäßig auf der Fläche verteilt. Die Leiterbahnen und andere Strukturen auf der Leiterplatte seien glatt. Gummiquetscher würden nicht gebraucht. Sobald der Farbstoff aufgetragen sei, werde er durch Bestrahlung mit einem Scanner aufgehärtet, wodurch die Bestandteile im Farbstoff aktiviert und in die endgültige, leitende und fest haftende Legierung umgewandelt würden. Gedruckt werde mit Farbstoffen unterschiedlicher Zusammensetzung. In einem einzigen Verfahrensdurchlauf können Leiterplatten mit verschiedenen Strukturkompositionen, elektrischen Eigenschaften und Stoffen, Speichern und sogar supraleitenden Elementen erzeugt werden.

Quelle: Blick durch die Wirtschaft vom 7. 8. 1989 Fa

BASIC-Programmbausteine

Lineare Gleichungssysteme, nichtlineare Gleichungen, Differentialgleichungen
von H. Rothe u. M. Steinbach, Reihe Technische Informatik, VEB Verlag Technik, Berlin 1989, 256 S., 39 Bilder, 9 Tafeln, Broschur DDR 25,– M, Bestell-Nr. 554 003 5

Basic ist auch bei Naturwissenschaftlern und Ingenieuren recht beliebt. Die Sprache ist einfach und gestattet eine schnelle Programmentwicklung. Für einen effizienten Einsatz benötigt man aber vorgefertigte Programmbausteine für häufig wiederkehrende Aufgaben. Hier schließt der Titel von Rothe und Steinbach eine Lücke.
Einleitend werden zunächst die Grundzüge von Basic zusammengefaßt dargestellt. Die Autoren wollen dadurch kein Lehrbuch ersetzen; dem bereits informierten Leser werden aber die Sprachkonstrukte ins Gedächtnis zurückgerufen. Besonders wertvoll erscheinen dabei die Darlegungen über die Genauigkeit numerischer Berechnungen, aber auch die Überlegungen zum Rechenzeitbedarf. Es folgen 75 Programme für drei klassische Gebiete der numerischen Mathematik: zur linearen Algebra und Matrizenrechnung, über nichtlineare Gleichungen und Gleichungssysteme sowie zur Lösung von Differentialgleichungen. Auch hier sind die wiederholten Bewertungen der Programme hervorzuheben, die nicht nur Fragen von Speicherplatz- und Laufzeitbedarf, sondern auch das Fehlverhalten betreffen. Der Rezensent sieht hierin einen besonderen Wert dieser Sammlung, die keinen Vergleich mit renommierten Standardwerken auf diesem Gebiet zu scheuen braucht. Die angegebenen Programme lassen sich übrigens wegen der nahen Verwandtschaft der Sprachen auch leicht in Fortran umsetzen.
Der Anhang gibt einen tabellarischen Überblick über Anweisungen und Funktionen von Basic. Interessant sind die Zeitvergleiche zwischen verschiedenen Basic-Systemen sowie die Benchmarktests zu Genauigkeit und Rechenzeitbedarf für eine Reihe von Sprachverarbeitungssystemen.

Prof. Dr. Werner

Der PC im Netzwerk

Bestandteile – Standards – Kriterien
von O. R. Schreiber, Reihe Chip Wissen, Vogel Buchverlag, Würzburg 1989, 226 S., ISBN 3-8023-0262-1, 40,– DM

Trotz der unbestrittenen Vorzüge der PC-Technik (günstiges Preis-Leistungs-Verhältnis, umfangreiche Standardsoftware, Einsatz am Arbeitsplatz) hatten zentrale Multiuser-Systeme dieser immer noch etwas voraus: den gemeinsamen Zugriff auf Ressourcen und die Programmentwicklung im Team. Durch Vernetzung kann der PC jetzt gleichziehen. Ja, mehr noch: Die verteilte Intelligenz sichert jedem Nutzer seinen Komfort auch bei stark expandierenden Netzen, die Produkte verschiedener Hersteller können einbezogen werden, eine flächendeckende Vernetzung (Wide Area Network) ist in Sicht. Gleichzeitig bedeutet es für immer mehr Anwender die Berührung mit dieser Problematik, unabhängig davon, ob sie aus dem Großrechner- oder PC-Bereich kommen.
Trotz der Informationsvielfalt ist in dem Buch eine umfassende, aber auch ohne Vorkenntnisse auf diesem Gebiet verständliche Darstellung gelungen. Jede Abkürzung wird ausgeschrieben, jeder Begriff erklärt. Zumeist handelt es sich aber – und das macht die spezifische Problematik der Materie aus – um Firmen- oder Produktnamen bzw. Standards. Gerade angesichts der zahlreichen firmenspezifischen Entwicklungen kommt der Darstellung der internationalen Standardisierungsbemühungen besondere Bedeutung zu. Der Autor widmet sich diesen Themen ausführlich; das OSI-Referenzmodell (Open Systems Interconnection) der ISO und IBMs Systems Network Architecture (SNA) werden gegenübergestellt.
Allgemeine Grundlagen zu lokalen Netzen beinhalten Topologien (Stern-, Ring-, Busnetze), Zugriffsverfahren und Protokolle. Der Leser erfährt, daß schon mit der Topologie über spätere Eigenschaften des Netzes (Erweiterbarkeit, Kabellänge) entschieden wird. Als wichtige Zugriffsverfahren werden der IBM-Token-Ring und das CSMA/CD-Protokoll, das dem Ethernet-Standard zugrunde liegt, gegenübergestellt. Beide Verfahren bilden die Grundlage für das Gros der kommerziellen Netzlösungen. Für die Transportprotokolle stehen NetBIOS, das von IBM als Standard etabliert wurde und auf das fast sämtliche Netzsoftware aufsetzt, und SPX/IPX von Novell.
Breiten Raum nimmt die Beschreibung von Codierungs- und Signalübertragungsverfahren ein – zu empfehlen all jenen, die es ob der trockenen Fachbücher schon aufgeben wollten, zu verstehen, was eigentlich auf der Leitung passiert. Der Exkurs in die Informationstheorie mag manchem Anwender zu theoretisch erscheinen, ist aber sehr anschaulich. Einige der erläuterten Begriffe (Entropie, Redundanz) sind nicht nur für Netzwerke relevant; die bekannten Kompressionsprogramme (PKARC) bedienen sich bespielsweise dieses Wissens.
Für Leser, die beabsichtigen, ein lokales Netz zu kaufen oder zu installieren, wird ein besonderer Service geboten: Checklisten mit Kriterien für die Bewertung von Multiuser-Betriebssystemen, für die Planung des Aufbaus eines LANs, Kostenanalyse und Einschätzung von Angeboten. Sie werden bemerken, daß es bei der Bewertung von lokalen Netzen nicht nur auf Geschwindigkeit und die Anzahl der Stationen ankommt, sondern auch auf Erweiterbarkeit, Kompatibilität zu anderen Produkten und Komfort, vor allem aber auf die Sicherheit des Gesamtsystems. Sicherheitskonzepten ist deshalb auch ein eigenes Kapitel gewidmet. Beschrieben werden von Novell Netware bekannte Verfahren: doppelte Directories und FATs, gespiegelte und duplizierte Festplatten, doppelte Fileserver, unterbrechungsfreie Stromversorgung, Transaction Tracking, Locking-Mechanismen und Paßwortschutz. Ein weiteres Kapitel beschreibt Verfahren zur Verbesserung der Performance (z.B. Caching). Warum aber findet sich kein Hinweis auf Netware, wo alle diese Techniken implementiert und ausführlich beschrieben sind?
Über seine Funktion als Einführung in das Gebiet der Rechnernetze hinaus kann das Buch auch später als Nachschlagewerk benutzt werden. Ein Glossar, ein ausführlicher Index und das bei dieser Reihe bewährte Absetzen der Begriffe auf dem Rand der Textseiten unterstützen ein schnelles Auffinden von Stichwörtern. Insgesamt ein überzeugendes Buch zu dieser Thematik.

U. Schulze

Wissensspeicher Prozeßrechentechnik

von einem Autorenkollektiv unter der Leitung von D. Balzer und J. Hesse, VEB Fachbuchverlag, Leipzig 1989, 1. Auflage, 376 S., 159 Bilder, 80 Tabellen, 11 Tafeln, DDR 36,– M

Ein Wissensspeicher nimmt eine Zwischenstellung zwischen einem Lehrbuch und einem Nachschlagewerk, z.B. in Form eines Lexikons, ein. Er stellt nach Sachgebieten geordnet das verfügbare Wissen in gedrängter Form bereit. Wegen der Kürze der Darstellung sollte er auf weiterführende Literatur verweisen. Die Autoren des Wissensspeichers Prozeßrechentechnik sind auch nach diesem Prinzip verfahren, jedoch wurden sämtliche Zeitschriftentitel abgekürzt und ohne Angabe des Verlages aufgenommen, so daß sie nur dem Fachmann zugänglich bleiben. Das widerspricht allerdings dem im Vorwort erhobenen Anspruch auf allgemeinverständliche Darstellung, mit der ein Leserkreis vom Meister bis zum Hochschulkader angesprochen werden soll. Die Verwendung vieler Fachbegriffe (ohne sie an gleicher Stelle zu erläutern) sowie die überwiegend wissenschaftliche Darstellungsweise lassen doch eher auf die geübte Lesergruppe Studenten bzw. Hoch- und Fachschulabsolventen schließen.
Der Wissensspeicher stellt eine weitgehend vollständige Sammlung von Definitionen und Formeln insbesondere aus den Gebieten Aufbau und Wirkungsweise von Rechnern, Programmsysteme, Zuverlässigkeit, Meßwerterfassung und -verarbeitung, Systemanalyse, Regler und Optimierung dar. Durch seine von früheren Wissensspeichern übernommene übersichtliche Aufmachung lassen sich gesuchte Sachgebiete schnell auffinden. Die Punkte 1 bis 3 geben zwar vor, sich mit der Prozeßrechnerproblematik zu befassen, gelten aber auch für andere Rechner. Ähnliches trifft auch auf weitere Kapitel, in denen die Spezifik der Prozeßrechner gegenüber anderen Rechnern nicht herausgestellt wird, zu. Gänzlich habe ich irgendwelche Bezüge zu CAM oder CIM vermißt. Ursache dafür dürfte ein viel zu langer Druckvorlauf dieses Buches sein, denn die jüngsten zitierten Zeitschriften – in denen bekanntlich neues Wissen zuerst veröffentlicht wird – sind aus dem Jahre 1984.

H. Hemke

Datenbank – Engineering mit Clipper

von M. Zwernemann u. H.-P. Förster, Vogel Buchverlag Würzburg, Würzburg 1989, 461 S., ISBN 3-8023-0263-x

Der Titel des Buches verrät schon, daß es sich von herkömmlichen Clipper-Büchern unterscheidet. Das Nachschlagewerk umfaßt 461 Seiten und ist in 4 Teile gegliedert. Etwa 100 Seiten des Buches sind dem Clipper-Engineering vorbehalten. Dabei liegt der Schwerpunkt auf der Programmplanung, der Vorbereitung und der Programmierung einer Adreßverwaltung für Kleinbetriebe.
Von den Autoren wird die Bedeutung des Clipper-Softwarepaketes als eigenständiges Entwicklungswerkzeug hervorgehoben. Das Buch beginnt mit grundsätzlichen Ausführungen zum Aufbau einer Datenbank. Am Beispiel einer Kundendatei wird das Clipper-Programm DBU (dBase-Utilities) erläutert. Die Beschreibung des Programms DBU ist gut verständlich. Selbstverständlich können mit Clipper-Programmen auch dBASE-Dateien bearbeitet werden.
Aber offenbar hatten die Autoren mehr die Absicht, die Selbständigkeit von Clipper hervorzuheben, als auf Gemeinsamkeiten und Unterschiede zum weitverbreiteten Datenbanksystem dBASE einzugehen. Vermißt wurde leider ein Hinweis auf die Verwendbarkeit von dBASE-Indexdateien, denn diese unterscheiden sich nicht nur durch die Erweiterung .NDX, sondern sind auch wesentlich langsamer als Clipper-Indexdateien vom Typ .NTX.
Es folgt ein Kurzporträt des Clipper-Compilers einschließlich seiner Installation auf der Festplatte. Dies ist wohl der interessanteste Teil des Buches, da er nicht nur den Compiler und Linker behandelt, sondern auch auf die Fehlersuche mit dem Clipper-Debugger eingeht.
Teil II des Buches befaßt sich umfangreich mit Clipper-Engineering. Breiten Raum nehmen die Voranalyse, Grobkonzeption und Programmvorbereitung ein. Diese theoretischen Aussagen werden dann durch konkrete Listings untersetzt und können mit einer zum Lieferumfang gehörenden Beispieldiskette erprobt werden. Schließlich findet der Leser in Teil III und IV eine umfassende Dokumentation aller Clipper-Befehle und Funktionen. Zweck, Syntax und Optionen der Befehle sowie Hinweise auf die zu verwendende Library werden in übersichtlicher Form mit kleinen Programmbeispielen erläutert. Alles in allem ein Buch, das in der Praxis als Nachschlagewerk schnell weiterhilft und seinem Anspruch auf vollständige Behandlung des Clipper-Softwarepaketes gerecht wird.

J. Richter

Bemerkung: Für die Funktionen von Clipper gelten die gleichen Vereinbarungen wie für die Kommandos (siehe MP 3/90). Der Ergebnistyp der Funktionen ist folgendermaßen abgekürzt: S: String, N: numerisch, I: integer; L: logisch, D: Datum, M: Memo, W: wie <ausdr>, E: wie Eingabe.
Unter den Funktionen der Clipper-Sprache sind einige, die keinen Ergebniswert liefern. Ihr Ergebnistyp wird hier durch einen Strich gekennzeichnet. Kommas werden auch als Platzhalter in Parameterlisten verwendet. Sie können entfallen, wenn ihnen kein Parameter folgt.

Funktion und Aufruf	Typ	Bemerkung
& <speichervar>	S	Makroersetzung
ABS (<n_ausdr>)	N	absoluter Wert
ACHOICE (<oben>, <links>, <unten>, <rechts>, <feld_1>[,<feld_2>[,<funktion>[,<n_ausdr_1>[,<n_ausdr_2>]]]])	N	Eingabe über Pop-up-Menü
ACOPY (<quell_feld>,<ziel_feld>[,<anfang_in_quelle>[,<anzahl>[,<anfang_im_ziel>]]])	–	Feldinhalt kopieren
ADEL (<feld>,<index>)	–	Feldinhalt löschen
ADIR (<filegruppe>[,<namen_feld>[,<größen_feld>[,<datum_feld>[,<zeit_feld>[,<attribut_feld>]]]]])	N	DOS-Verzeichnis in Felder übertragen
AFIELDS (<namen_feld>[,<typ_feld>[,<längen_feld>[,<dez_feld>]]])	N	Felder mit Feldinformationen füllen
AFILL (<feld>,<wert>[,<anfang>[,<anzahl>]])	–	Feld mit einem Wert initialisieren
AINS (<feld>,<position>)	–	Element in ein Feld einfügen
ALIAS (<n_ausdr>)	S	Name des aktuellen Arbeitsbereiches
ALLTRIM (<c_ausdr>)	S	alle Leerzeichen entfernen
ALTD ([<n_ausdr>])	–	Debugger aufrufen Alt-D ein/aus
AMPM (<c_ausdr>)	S	Zeitangabe in am/pm-Form umwandeln
ASC (<c_ausdr>)	N	Kodewert des ersten Zeichens
ASCAN (<feld>,<n_ausdr>[,<anfang>[,<anzahl>]])	N	Position eines Wertes in einem Feld
ASORT (<feld>[,<anfang>[,anzahl>]])	–	Feld aufsteigend sortieren
AT (<c_ausdr_1>,<c_ausdr_2>)	N	Position von <c_ausdr_1> in <c_ausdr_2>
BIN2I (<c_ausdr>)	N	Konvertierung 2-Byte-Integer in numerisch
BIN2L (<c_ausdr>)	N	Konvertierung 4-Byte-Integer in numerisch
BIN2W (<c_ausdr>)	N	Konvertierung 2-Byte-CARDINAL in numerisch
BOF ( )	N	Fileanfang
CDOW (<d_ausdr>)	S	Wochentag
CHR (<n_ausdr>)	S	Zeichen mit dem Wert <n_ausdr>
CMONTH (<d_ausdr>)	S	Name des Monats
COL ( )	N	aktuelle Kursorspalte
CTOD (<c_ausdr>)	D	Konvertierung String in Datum
CURDIR ([<gerät>])	S	aktueller DOS-Pfad auf <gerät>
DATE ( )	D	aktuelles Datum
DAY (<d_ausdr>)	N	Tag des Monats
DAYS (<n_ausdr>)	N	Sekunden in Tage konvertieren
DBEDIT ([<oben>,<links>[,<unten>,<rechts>]],[<feld_1>],[<c_ausdr>],[<feld_2>],[<feld_3>],[<feld_4>],[<feld_5>],[<feld_6>],[<feld_7>])	–	Editieren von Sätzen aus mehreren Arbeitsbereichen in einem Fenster
DBF ( )	S	Name des aktuellen Datenbankfiles
DBFILTER ( )	S	Filterbedingung des aktuellen Arbeitsbereichs
DBRELATION (<n_ausdr>)	S	Verbindungsausdruck im aktuellen Arbeitsbereich
DBRSELECT (<n_ausdr>)	N	Zielarbeitsbereich einer Verbindung
DELETED ( )	L	Abfrage der Löschmarkierung
DESCEND (<ausdr>)	W	absteigende Ordnung beim Indizieren und Suchen
DISKSPACE ([<gerät>])	N	freier Speicherplatz auf <gerät>
DOSERROR ( )	N	Nummer des letzten DOS-Fehlers
DOW (<d_ausdr>)	N	Nummer des Wochentages
DTOC (<d_ausdr>)	S	Konvertierung D in S (tt.mm.jj)
DTOS (<d_ausdr>)	S	Konvertierung D in S (jjjjmmtt)
ELAPTIME (<zeit_1>,<zeit_2>)	S	Zeitdifferenz in der Form hh:mm:ss
EMPTY (<ausdr>)	L	Test auf leeren Ausdruck
EOF ( )	L	Ende des Datenbankfiles
ERRORLEVEL ([<n_ausdr>])	N	DOS-Fehlerniveau ermitteln/setzen
EXP (<n_ausdr>)	N	Exponentialfunktion
FCLOSE (<n_ausdr>)	L	DOS-File schließen
FCOUNT ( )	N	Anzahl der Felder im aktuellen Datenbankfile
FCREATE (<file>[,<attribut>])	N	DOS-File mit Attributen erzeugen
FERROR ( )	N	DOS-Fehlernummer nach Filezugriff
FIELD/FIELDNAME (<n_ausdr>)	S	Name des n-ten Feldes
FILE (<c_ausdr>)	L	Existenz eines Files
FKLABEL (<n_ausdr>)	S	Name der Funktionstaste
FKMAX ( )	N	Zahl der Funktionstasten
FLOCK ( )	L	ausschließlich Dateizugriff
FOPEN (<file>[,<zugriffsart>])	N	DOS-File eröffnen
FOUND ( )	L	Erfolg von Suchoperationen
FREAD (<dateikennung>,<puffer>,<anzahl>)	N	Bytes aus DOS-File lesen
FREADSTR (<dateikennung>,<anzahl>)	S	Bytes aus DOS-File lesen
FSEEK (<dateikennung>,<anzahl>,<modus>)	N	DOS-Funktion SEEK
FWRITE (<dateikennung>,<puffer>,<anzahl>)	N	Puffer in DOS-File ausgeben
GETE (<speichervar>)	S	Inhalt der DOS-Umgebungsvariablen
HARDCR (<c_ausdr>)	S	weiche in harte Zeilenwechsel wandeln
HEADER ( )	N	Kopflänge der aktuellen Datenbank
I2BIN (<n_ausdr>)	S	Konvertierung in 2-Byte-CARDINAL
IF/IIF (<bedingung>,<ausdr_1>,<ausdr_2>)	E	bedingter Ausdruck
INDEXEXT ( )	S	Namenserweiterung der Indexfiles
INDEXKEY (<n_ausdr>)	S	Schlüsselausdruck eines Indexfiles
INDEXORD ( )	N	Nummer des aktuellen Hauptindex in Liste
INKEY ( )	N	Kode einer Taste
INT (<n_ausdr>)	N	ganzzahliger Teil
ISALPHA (<c_ausdr>)	L	.T.: erstes Zeichen ist Buchstabe
ISCOLOR ( )	L	.F.: Monochrombildschirm
ISLOWER (<c_ausdr>)	L	.F.: erstes Zeichen Großbuchstabe
ISPRINTER ( )	L	.T.: Drucker (LPT1) ist bereit
ISUPPER (<c_ausdr>)	L	.T.: erstes Zeichen Großbuchstabe
L2BIN (<n_ausdr>)	S	Konvertierung N in 4-Byte-Integer
LASTKEY ( )	N	Kode der zuletzt gedrückten Taste
LASTREC ( )	N	Anzahl der Datensätze
LEFT (<c_ausdr>,<n_ausdr>)	S	linke Teilzeichenkette
LEN (<c_ausdr>)	N	Länge eines Strings
LENNUM (<n_ausdr>)	N	Zeichenkettenlänge eines Wertes
LOCK ( )	L	Sperren des aktuellen Satzes
LOG (<n_ausdr>)	N	Logarithmus
LOWER (<c_ausdr>)	S	Konvertierung in Kleinbuchstaben
LTRIM (<c_ausdr>)	S	Entfernen führender Leerzeichen
LUPDATE ( )	D	Datum der letzten Korrektur
MAX (<n_ausdr_1>,<n_ausdr_2>)	N	Maximum
MEMOEDIT (<cm_ausdr>[,<oben>,<links>,<unten>,<rechts>[,<flag>],[<funktion>],[<tab>],[<zeile>],[<spalte>],[<rel_zeile>][,<rel_spalte>]])	S, M	Editieren eines Memo-Feldes
MEMOLINE (<cm_ausdr>,[<anzahl>],[<zeile>],[<tab>][,<l_ausdr>])	S, M	formatierte Zeile
MEMOREAD (<file>)	S, M	Inhalt des Textfiles
MEMORY ( )	N	freier Hauptspeicher in KByte
MEMOTRAN (<cm_ausdr>,<zeichen_1>,<zeichen_2>)	S, M	Ersetzen der Zeilenendezeichen
MEMOWRIT (<file>,<cm_ausdr>)	L	Text in ein File schreiben
MIN (<n_ausdr_1>,<n_ausdr_2>)	N	Minimum
MLCOUNT (<c_ausdr>,[<anzahl>],[<tab>][,<l_ausdr>])	N	Anzahl der Zeichen
MLPOS (<c_ausdr>,<anzahl>,<zeile>)	N	Position einer Zeile im String
MOD (<n_ausdr_1>,<n_ausdr_2>)	N	Rest bei einer Division
MONTH (<d_ausdr>)	N	Nummer des Monats
NETERR ( )	L	Erfolg von Netz-Operationen
NETNAME ( )	S	Name eines Rechners im Netz
NEXTKEY ( )	N	Kode der nächsten gedrückten Taste ohne Lesen
OS ( )	S	Name des Betriebssystems
PCOL ( )	N	Spaltenposition des Druckkopfes
PCOUNT ( )	N	Anzahl der Parameter in der Kommandozeile des Betriebssystems
PROCLINE ( )	N	Zeilennummer im Quellkode
PROCNAME ( )	S	aktuelle Prozedur
PROW ( )	N	Zeilenposition des Druckkopfes
RAT (<c_ausdr_1>,<c_ausdr_2>)	N	letztes Auftreten von <c_ausdr_1> in <c_ausdr_2>
READEXIT (<l_ausdr>])	L	Pfeiltasten ↑ ↓ beenden READ
READINSERT ([<l_ausdr>])	L	Einfügemodus in READ ein/aus
READKEY ( )	N	Kode der Taste, mit der READ verlassen wurde
READVAR ( )	S	Name der zu lesenden Variablen
RECCOUNT ( )	N	Anzahl der Datensätze
RECNO ( )	N	Nummer des aktuellen Satzes
RECSIZE ( )	N	Länge eines Datensatzes
REPLICATE (<c_ausdr>,<n_ausdr>)	S	Zeichenkette vervielfachen
RESTSCREEN (<oben>,<links>,<unten>,<rechts>,<c_ausdr>)	–	Rückspeichern des geretteten Fensters
RIGHT (<c_ausdr>,<anzahl>)	S	rechte Teilzeichenkette
RLOCK ( )	L	Sperren des aktuellen Satzes
ROUND (<n_ausdr>,<dez>)	N	Runden
ROW ( )	N	aktuelle Zeile auf dem Bildschirm
RTRIM ( )	S	Entfernen nachfolgender Leerzeichen
SAVESCREEN (<oben>,<links>,<unten>,<rechts>)	S	Fensterinhalt retten
SCROLL (<oben>,<links>,<unten>,<rechts>,<anzahl>)	–	Fenster verschieben
SECONDS ( )	N	Sekunden seit Mitternacht
SECS (<c_ausdr>)	N	Konvertierung Zeit in Sekunden
SELECT ( )	N	Nummer des Arbeitsbereichs
SETCANCEL ([<l_ausdr>])	L	Programmabbruch mit [ALT]-C ein/aus
SETCOLOR ([<c_ausdr>])	S	Farbeinstellung oder -abfrage
SETPRC (<prow>,<pcol>)	–	Einstellen Druckzeile und -spalte
SOUNDEX (<c_ausdr>)	S	phonetische Konvertierung
SPACE (<n_ausdr>)	S	<n_ausdr> Leerzeichen
SQRT (<n_ausdr>)	N	Quadratwurzel
STR (<n_ausdr> [,<länge>[,<dez>]])	S	Konvertierung Wert in String
STRTRAN (<c_ausdr>,<alt>,[<neu>],[<n_ausdr>][,<anzahl>])	S	Zeichenkette transformieren
STRZERO (<n_ausdr> [,<länge>[,<dez>]])	S	wie **STR** mit führenden Nullen
STUFF (<c_ausdr>,<position>,<anzahl>,<neu>)	S	Teilzeichenkette ersetzen
SUBSTR (<c_ausdr>,<start>,<länge>])	S	Substring
TIME ( )	S	Systemzeit
TONE (<frequenz>,<dauer>)	–	akustisches Signal ausgeben
TSTRING (<n_ausdr>)	S	Konvertierung sec in Zeit (hh:mm:ss)
TRANSFORM (<ausdr>,<maske>)	S	Konvertierung entsprechend Maske
TRIM (<c_ausdr>)	S	Entfernen nachfolgender Leerzeichen
TYPE ("<c_ausdr>")	S	Typ eines Bezeichners
UPDATED ( )	L	.T.: Daten wurden bei READ verändert
UPPER (<c_ausdr>)	S	Konvertierung in Großbuchstaben
USED ( )	L	.T.: Arbeitsbereich ist aktuell
VAL (<c_ausdr>)	N	Konvertierung in numerischen Wert
VERSION ( )	S	Clipper-Version
WORD (<n_ausdr>)	I	Konvertierung in 16-Bit-Integer
YEAR (<d_ausdr>)	N	Angabe des Jahrhunderts

Zusammengestellt von Dr. Wilfried Grafik

ISSN 0232-2892
Mikroprozessortechnik, Berlin 4 (1990) 4
S. 97–128

vorgestellt

PizazzPlus

Die erweiterte Hardcopy-Funktion

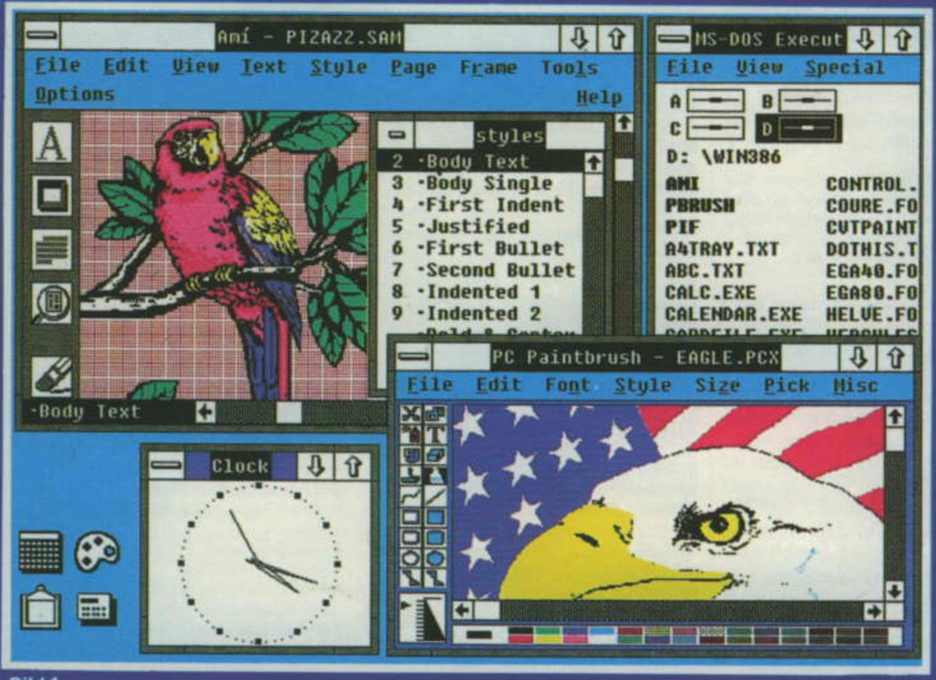


Bild 1

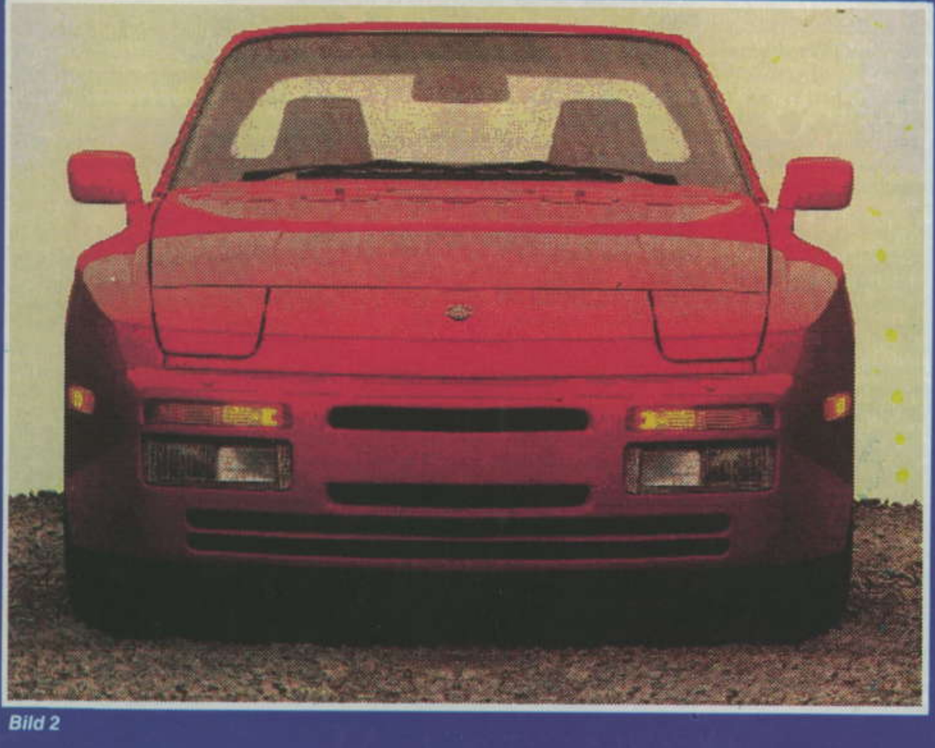


Bild 2

Wer auf dem PC an eine Print-Screen-Taste zum Sofortausdruck des Bildschirminhalts gewöhnt ist, wird mit Interesse die Vorstellung des menügeführten PizazzPlus (Application Techniques, Inc.) in der deutschen Version 1.3 (P. Rosenthal GmbH) auf der Systems 89 zur Kenntnis genommen haben. Und 32 KByte im RAM für das residente Programm dürften auch auf jedem der geeigneten Computer noch vorhanden sein. Die Taste PrtSc (auch die deutsche Tastenbezeichnung *Druck* ist in diesem Falle etwas treffender) aktiviert dann ein richtiges menügeführtes Druckprogramm mit einer ganzen Reihe von Gestaltungsmöglichkeiten, die weit hinausgehen über den profanen Ausdruck des Bildschirminhalts.

Als Hardwarevoraussetzungen sind ein PC XT/AT oder ein Kompatibler, das IBM PS/2, aber auch ein Toshiba T1600, T3200, T5100 und andere sind geeignet. Haben Sie Ihren Computer in der angebotenen Liste gefunden, setzen Sie die Installation mit der Wahl des Bildschirms und des Druckers fort. Sie können aus mehr als 20 Monitortypen und aus über 300 Druckern die an Ihren Computer angeschlossenen auswählen. Die weit verbreiteten (Farb-) Grafik-Adapter CGA, VGA, EGA und Hercules sind in den Listen ebenso enthalten wie die Drucker LX-86 und LX-800. Die vollen Möglichkeiten des Ausdruckens, wie sie in den beiden Bildern zu sehen sind, bieten aber erst Farbdrucker mit der Qualität beispielsweise des HP PaintJet Color von Hewlett Packard. Wie im Bild 1 zu sehen ist, können die Bilder und Grafiken vergrößert und verkleinert werden; die Position des ganzen Bildes oder aber eines Ausschnittes können Sie sowohl auf dem Bildschirm als auch auf dem Papier beim anschließenden Druck ändern, wenn Ihnen der im Draft-Modus ausgedruckte Entwurf nicht gefällt. Nach diesem Druck werden die Halbtöne, Graustufen und Farben eingestellt und der Druck im *Präsentationsdruck* mit einer höheren Auflösung der Grau- bzw. Farbstufen und größerem Kontrast wiederholt. PizazzPlus wird ja, wie oben beschrieben, für Bildschirm und Drucker installiert, so daß Sie auch von einem Schwarzweiß-Bildschirm Farbausdrucke erhalten können und umgekehrt. Die Farben sind in einem Fenster mit dem Namen oder der Nummer dargestellt und können nach dem mitgelieferten Druckmusterkatalog geändert werden. Und da wir Ihnen ja hier ein Programm vorstellen wollen, werden Sie mit Recht vermuten, daß nicht nur die Bearbeitung und der Ausdruck möglich sind, sondern auch die Speicherung der gewonnenen Daten und deren Weiterverarbeitung mit einem Textverarbeitungs- oder auch mit einem Desktop-Publishing-System. Dateien können unter anderem von Word, Wordperfekt, Microsoft-Windows, Pagemaker und dem Ventura-Publisher übernommen und in deren Dokumente eingesetzt werden. *MP*

ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

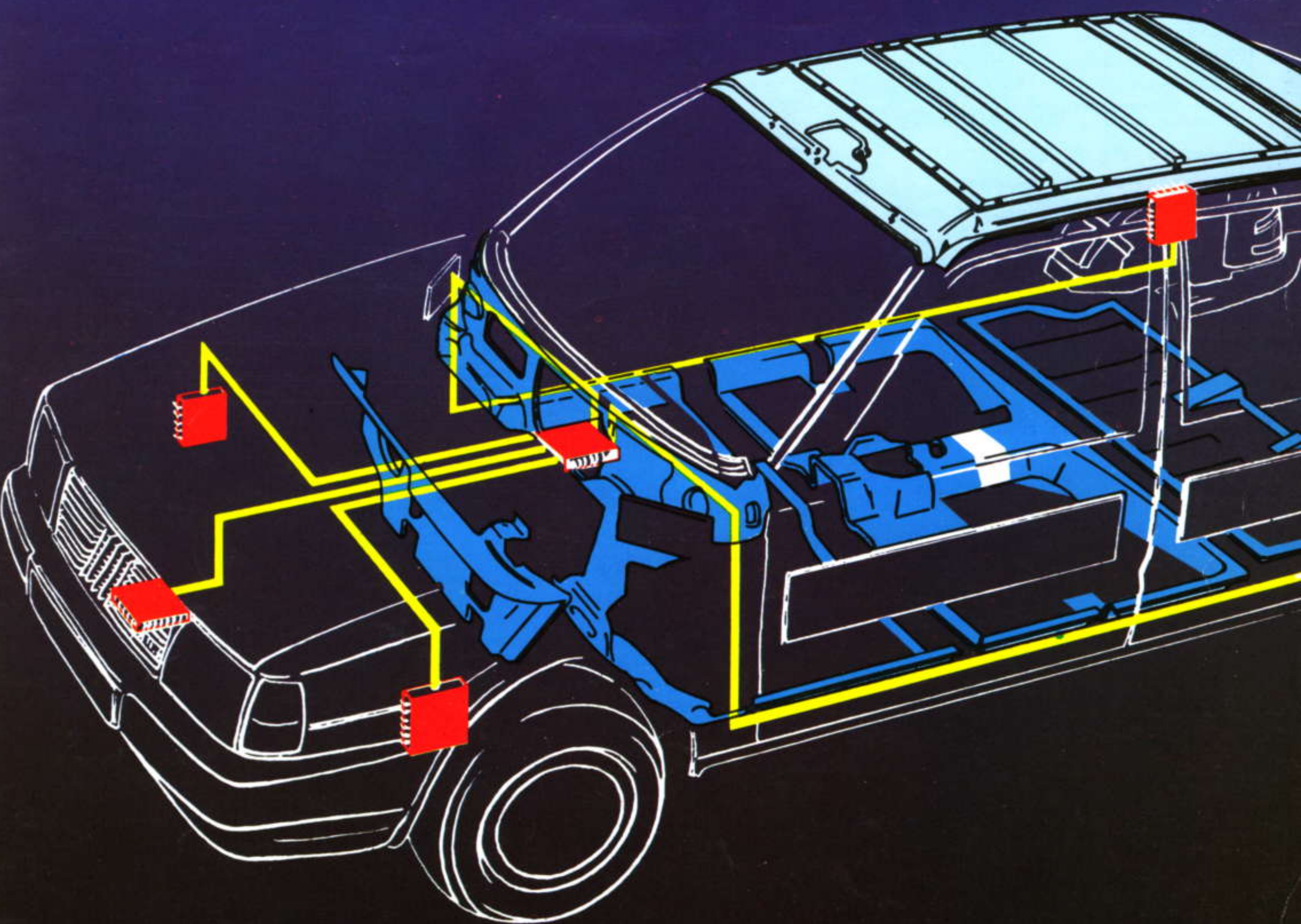
Heft 5 · 1990

MP Mikroprozessortechnik

VEB Verlag Technik Berlin

ISSN 0232-2892

mit Beiträgen aus c't
und Computer persönlich



Mikrocontroller

Referenzkarte C

Datentypen und Typdefinition

Grunddatentypen 15

char
int, short [int], long [int], unsigned [int]
float, double (= long float)

Anmerkung: Die Seitenangaben beziehen sich auf das im VEB Verlag Technik erschienene Buch „Programmieren mit C" von M. Clauß und G. Fischer

Kombinierte Typen (nicht für alle Compiler) 16

unsigned short, unsigned long, unsigned char

Strukturtyp 53

```
struct [<strukturtypbezeichner>] {
   <typ> <bezeichner>;
      ...
}
```

Bitfeldtyp 56

```
struct [<strukturtypbezeichner>] {
   [unsigned <bezeichner>]: <konstantenausdruck>IO;
      ...
}
```

Uniontyp 58

```
union [<uniontypbezeichner>] {
   <typ> <bezeichner>;
      ...
}
```

Aufzählungstyp 60

```
enum [<aufzählungstypbezeichner>] {
   <enum-bezeichner> [= <konstantenausdruck>] [, ...]
}
```

Definition von synonymen Typbezeichnern 61

```
typedef <typ> <bezeichnerliste>;
   typedef struct datum DATUM;
```

Konstanten

Integerkonstanten 17

dezimal :	**12**	**12L**	**65567**
oktal :	**012**	**012L**	**0777777**
hexadezimal:	**0x12**	**0x12L**	**0xFFFFF**

Realkonstanten 17

1E3	**1e3**	**1.e−3**
.47	**2.**	**−12.34e−6**

Zeichenkonstanten (int) 17

'A' **'\n'** **'\011'**

Zeichenkettenkonstanten 18

"a" **" "** **"abcdefg"**

Variablen

```
[<speicherklasse>] <typ> <bezeichnerliste>;
int i, *ptr, k[5] = { 1, 3, 18 };
static char b[ ] = "Programm", a[5] [2];
static char *keyword[ ] = { "if", "else", ..., "for" };
struct datum heute;
struct {
   short flag;
   char string[80];
} table[25];
```

Implizite Typkonvertierungen

Funktionsargumente und Operanden arithmetischer Ausdrücke

char, short	**-->**	**int**
unsigned char, unsigned short	**-->**	**unsigned int**
float	**-->**	**double**

Teilausdrücke verschiedener Typen in arithmetischen Ausdrücken

unsigned int -- long -- unsigned long -- double

Wertzuweisung und Rückgabe von Funktionswerten

Konvertierung in den Typ von <lvalue> ∕ der Funktion

Operatoren und Ausdrücke

Operator	Bezeichnung	Ausdruck	Assoz.
()	Funktionsruf	<einf_Ausdr>([<arg_liste>])	L
[]	Feldoperator	<einf_Ausdr>[<ausdruck>]	
−>	Strukturoperator	<einf_Ausdr> −> <bezeichner>	
.	Strukturoperator	<lvalue> . <bezeichner>	
~	Negation bitweise	~ <ausdruck>	R
!	Negation logisch	! <ausdruck>	
+ +	Inkrement	+ + <lvalue> | <lvalue> + +	
− −	Dekrement	− − <lvalue | <lvalue> − −	
−	Vorzeichen	− <ausdruck>	
(<typ>)	Typkonvertierung	(<typ>) <ausdruck>	
*	Zeigeroperator	* <ausdruck>	
&	Adreßoperator	& <lvalue>	
sizeof	Größenoperator	sizeof <ausdruck> | (<typ>)	
*	Multiplikation	<ausdruck> * <ausdruck>	L
∕	Division	<ausdruck> ∕ <ausdruck>	
%	Divisionsrest	<ausdruck> % <ausdruck>	
+	Addition	<ausdruck> + <ausdruck>	L
−	Subtraktion	<ausdruck> − <ausdruck>	
<<	Linksverschiebung	<ausdruck> << <ausdruck>	L
>>	Rechtsverschiebung	<ausdruck> >> <ausdruck>	
<	Vergleich	<ausdruck> < <ausdruck>	L
>	Vergleich	<ausdruck> > <ausdruck>	
< =	Vergleich	<ausdruck> < = <ausdruck>	
> =	Vergleich	<ausdr> > = <ausdr>	
= =	Vergleich	<ausdruck> = = <ausdruck>	L
! =	Negation, ungleich	<ausdruck> ! = <ausdruck>	
&	UND bitweise	<ausdruck> & <ausdruck>	L
^	EXOR bitweise	<ausdruck> ^ <ausdruck>	L
|	ODER bitweise	<ausdruck> | <ausdruck>	L
& &	UND logisch	<ausdruck> & & <ausdruck>	L
||	ODER logisch	<ausdruck> || <ausdruck>	L
? :	Entscheidung	<ausdr> ? <ausdr> : <ausdr>	R
=	Wertzuweisung	<lvalue> = <ausdruck>	R
<op> =	zusammengesetzte Zuweisung	<lvalue> <op> = <ausdruck> <op> : + − * ∕ % & | ^ << >>	
,	Kommaoperator	<ausdruck> , <ausdruck>	L

Achtung:
Die Reihenfolge der Auswertung der Operanden binärer Operatoren ist nicht definiert! Logische Ausdrücke werden nur so weit wie nötig berechnet! L: linksassoziativ, R: rechtsassoziativ

Speicherklassen für Variablen und Funktionen

Variablen- und Funktionsdeklaration	Gültigkeitsbereich	Speicherklasse implizit	Speicherklasse erlaubt	Lebensdauer	Speichervergabe/ Codeerzeugung	Initialisierung implizit	Initialisierung explizit (~Unions!)	Anmerkung
Variablen innerhalb von Funktionen oder Blöcken	Block	auto	auto	Block	dynamisch	unbestimmt	Code-erzeugung	keine explizite Initialis. für Felder/Strukt.
			register	Block	Register	unbestimmt	Code-erzeugung	nicht alle Typen, nicht adressierbar
			extern	–	nein	–	–	Importvariable oder weiter unten definiert
			static	Programm	statisch	0	Compiler	
Variablen außerhalb von Funktionen oder Blöcken	von der Definition bis zum Ende des Quelltextes	extern	extern, implizit	Programm	statisch	0	Compiler	Exportvariable
			extern, explizit	–	nein	–	–	Importvariable
			static	Programm	statisch	0	Compiler	kein Export
Funktionsparameter		auto	auto, implizit	Block	dynamisch	Wert	–	
			register	Block	Register	Wert	–	
Funktionen außerhalb von Funktionen oder Blöcken	Punkt der Definition bis Ende Quelltext	extern	extern Deklaration	–	nein	–	–	Importfunktion oder weiter unten definiert
			extern Definition	–	ja	–	–	Exportfunktion
			static	–	ja	–	–	kein Export
Funkt. innerhalb von Funktionen oder Blöcken	Block	extern	extern	–	nein	–	–	Importfunktion oder weiter unten definiert

Bezeichnerklassen

Variablennamen, Funktionsparameter, Enum-Elemente	Funktionsnamen Typedef-Typbezeichner

Strukturtypen, Aufzählungstypen	Uniontypen

Struktur-komponenten	Struktur-komponenten	. . .	Struktur-komponenten
Union-komponenten	Union-komponenten	. . .	Union-komponenten

Die Länge von Bezeichnern ist nicht begrenzt, aber nur die ersten 8 Zeichen sind signifikant.

Funktionen

Definition:

```
[static] <typ> <funktionsname> <parameterliste>)
   <parameterdeklarationsliste>
<block>
```

Deklaration:

```
[extern] <typ> <funktionsname> ( );
```

Typen von Funktionsparametern:
- **alle Datentypen (Wertübergabe)**
- **Zeiger auf beliebige Datentypen, Felder und Funktionen; Felder, Zeichenkettenkonstanten und Funktionen werden als Zeiger übergeben**
- * **Die Funktion muß in der rufenden Funktion deklariert sein.**

Typen von Funktionsergebnissen:
- **alle Datentypen, auch Typattribut void**
- **Zeiger auf beliebige Datentypen, Felder und Funktionen**
- * **Die Funktionsabarbeitung endet mit der Returnanweisung oder Erreichen des Blockendes.**

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
4. Jahrgang · ISSN 0232–2892

Mikroprozessortechnik, Heft 5 · 90

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag Verlag Technik, Oranienburger Str. 13/14, DDR–1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Hans Weiß, Verantwortlicher Redakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2 87 02 03), Hans-Joachim Hill (Tel. 2870209); Sekretariat Tel. 2870381

Gestaltung Christina Bauer

Titel Frank-Norbert Beyer

Beirat Prof. Dr. sc. Thomas Horn, Prof. Dr. Bernd Junghans, Dr. Dietmar Keller, Prof. Dr. sc. Bernd-Georg Münzer, Prof. Dr. sc. Peter Neubert, Prof. Dr. sc. Dr. Michael Roth (Vorsitzender), Dr. Gerhard Schulze, Prof. Dr. sc. Manfred Seifart, Dr. Dieter Simon, Prof. Dr. sc. Dr. Jürgen Zaremba

Lizenz-Nr. 1710 des Presse- und Informationsdienstes der Regierung der DDR

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

Anzeigenannahme Verlag Technik, Anzeigenverwaltung, PSF 201, Berlin, 1020; Tel. 2870254, 2870291

Erfüllungsort und Gerichtsstand Berlin-Mitte. Der Verlag behält sich alle Rechte an den von ihm veröffentlichten Aufsätzen und Abbildungen, auch das der Übersetzung in fremde Sprachen, vor. Auszüge, Referate und Besprechungen sind nur mit voller Quellenangabe zulässig.

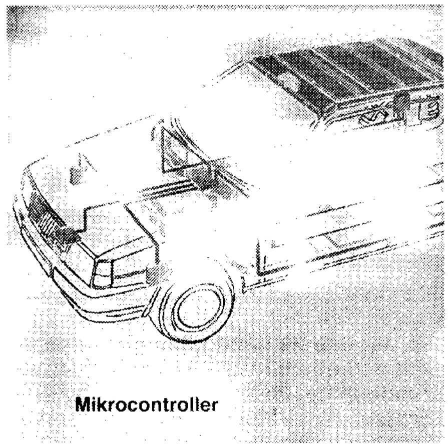
Redaktionsschluß 9. März 1990

AN (EDV) 49837

Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propagandititt te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustředni Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *Republik Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Republik Ungarn:* P.K.H.I., Külföldi Előfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nôi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR – 7010 Leipzig und Leipzig Book Service, Talstraße 29, DDR – 7010 Leipzig



Mikrocontroller

Mikrocontroller und Einchipmikrorechner für eingebettete Systeme dominieren auf dem Mikroprozessormarkt. Im Gerätebau, in der Prozeß- und Produktionsautomatisierung sowie in der Konsumgüterindustrie kommen z. Z. vor allem 8-Bit-Systeme zum Einsatz. 16- und 32-Bit-Systeme werden in den 90er Jahren bestimmend sein. Mit ASIC-Zellenbibliotheken werden Prozessorkerne und Peripheriebausteine nutzbar gemacht. Der Beitrag auf der **Seite 130** stellt die Architekturlinien des Marktführers Intel vor. Neben dem Schwerpunkt – dem MCS-51-System – werden die Systeme 80186, 80376, MCS-48, MCS-96 sowie 80960 behandelt.

Computer PERSÖNLICH

Sollte Ihnen unser Hinweis auf dem Titelbild entgangen sein, so möchten wir Sie an dieser Stelle darauf aufmerksam machen, daß Sie in der MP ab diesem Heft künftig Beiträge auch aus der Zeitschrift Computer Persönlich (CP) finden werden. Eine entsprechende Vereinbarung zum gegenseitigen Nachdruck von Artikeln wurde Anfang Februar zwischen den Redaktionen unterzeichnet. Die Beiträge der im Verlag Markt & Technik erscheinenden CP wenden sich vor allem an den beruflichen PC-Anwender, so daß wir sie als sinnvolle Ergänzung zu den Beiträgen aus der c't ansehen.
In diesem Heft beginnen wir mit der Vorstellung von drei Betriebssystemen, die mit Multitasking- und Multiuserfähigkeiten aufwarten wie etwa Unix, darüber hinaus jedoch MS-DOS-kompatibel sind (**Seite 141**).
Der zweite CP-Beitrag stellt in kurzer Form das neue Flaggschiff der Motorola-Mikroprozessoren, den 68040 vor, der mit 1,2 Millionen integrierten Transistoren und einer Leistung von 20 MIPS Intels 80486 Konkurrenz machen wird (**Seite 153**).

Vorschau

Für das Heft 6/1990 bereiten wir für Sie unter anderem Beiträge zu folgenden Themen vor:
- Protected Mode
- i80960

Schnittstellen von Turbo-C zu MS-DOS

Inhalt

Referenzkarte C 2. US

Gert-Ulrich Vack:
Mikrocontroller und Mikrorechner 130

Gabriele Buhren:
Fraktale als Interferenzeffekte 136

Wolfgang Thiel:
Eltern-Kind-Prozesse MS-DOS 137

Andreas Bliesener:
Grafik mit dem EC 1834 140

David Possin:
3 Systeme, 3 Methoden 141

MP-Kurs: 143
Manfred Zander:
Turbo-Pascal-Praxis (Teil 6)

Michael Wilde, Eckart Steffens:
Systeme mit System (Teil 3) 147

Peter Fölsche:
Berechnung einer zur Run-Time eingegebenen Funktion 150

68040-Prozessor: Power on the Chip 153

Wegbereiter der Informatik
Carl David Runge 153

MP-Computer Club 154
Eberhard Schmidt:
Typisierte CP/M-Dateien und Turbo-Pascal 5.0
Hartmut Voigt:
Hexadezimalverarbeitung

MP-Börse 156

Entwicklungen und Tendenzen 158

MP-Literatur 160

Mikrocontroller und Mikrorechner

Dr. Gert-Ulrich Vack, Dresden

In diesem Beitrag werden 8-, 16- und 32-Bit-Mikrocontroller und Mikroprozessoren für eingebettete Systeme vorgestellt. Dabei werden vor allem die 8-Bit-Familien MCS-48 und MCS-51 ausführlich diskutiert. Bisher besteht dazu in der DDR ein Informationsdefizit. Vor allem in Verbindung mit Entwurf, Herstellung und Applikation anwendungsspezifischer Schaltkreise erhalten diese 8-Bit-Mikrocontroller eine überragende Bedeutung für den Gerätebau.
Am Beispiel der Produktlinien von Intel wird zugleich ein Stück Firmenstrategie illustriert: Kompatibilität, Wiederverwendbarkeit und anwenderorientierter Zuschnitt der Produkte sind – neben anderen Faktoren – seit fast zwei Jahrzehnten eine Gewähr für Marktführerpositionen.

Was ist ein Mikrocontroller?

Der Geburtshelfer des Mikroprozessors, die US-amerikanische Firma Intel, führte bereits 1976 neben den bis dahin entwickelten 4- und 8-Bit-Mikroprozessoren 4004, 8008 und 8080 einen neuen Ast im Stammbaum der Mikroprozessoren ein: die Klasse der Mikrocontroller. Erster Repräsentant war die Familie MCS-48. Die Fortschritte im Integrationsgrad von Chips gestatteten es, neben einer 8-Bit-CPU noch I/O-Ports, RAM und ROM bzw. EPROM auf einem Chip zu vereinigen. Damit wurden völlig neue Anwendungsbereiche für Mikroprozessoren erschlossen, zum Beispiel in der Kfz-Elektronik, der Haushaltselektronik oder im großen Bereich der industriellen Elektronik (z. B. Maschinenbau). In der DDR hat sich für solche Prozessoren die international weniger gebräuchliche Bezeichnung *Einchipmikrorechner* eingebürgert.
Mit diesen Applikationen verbinden sich Begriffe wie *eingebetteter Computer* (englisch: embedded computer), *eingebetteter Controller* oder *eingebetteter Mikroprozessor*. Mit dem Adjektiv eingebettet wird die Besonderheit ausgedrückt, daß der Prozessor/Computer komplett im Gerät verschwindet, für den Anwender also nicht unmittelbar sichtbar ist. (Der Prozessor ist in seine Einsatzumgebung eingebettet.) Meist können nicht einmal mehr die Programme geändert werden, sie stehen nämlich im ROM bzw. im EPROM.
Eingebettete Computeranwendungen sind durch folgende typische Einsatzbedingungen charakterisiert:
– kompakte, wenig platzaufwendige, kostengünstige Hardware mit geringem Stromverbrauch und wenig Aufwand an „Rundumhardware" (englisch: glue logic, wörtlich „Logik zum Zusammenleimen")
– hohe Zuverlässigkeit auch unter ungünstigen Einsatzbedingung (z. B. Temperatur, Druck, harte Strahlung, Beschleunigungen, elektrische und magnetische Störfelder), Fehlertoleranz für besondere Anforderungen
– Reaktionsfähigkeit im Sinne eines Echtzeitsystems
– einfache, möglichst problembezogene Programmierung mit leistungsfähiger Programmierumgebung (z. B. Echtzeitdebugger, Emulator).

Der Kostenfaktor ist der bestimmende Aspekt, weil Geräte mit integriertem Mikrocontroller überwiegend in deutlich niedrigeren Stückzahlen verkauft werden als zum Beispiel Universal-PCs. (Anwendungen wie etwa die Kfz-Elektrik oder Haushaltgeräte sind die Regel bestätigende Ausnahmen.) Die Kosten eines eingebetteten Rechnersystems müssen auf deutlich weniger Installationen verteilt werden als bei massenhaft produzierten Universalrechnern.
Diese Forderungen sind nur erfüllbar, wenn die eingebetteten Rechner eine einfache und robuste Schaltungstechnik aufweisen und auf der Basis leistungsfähiger, funktionell kompakter Prozessorarchitekturen arbeiten.
Dem Merkmal der Echtzeitfähigkeit kommt besondere Bedeutung zu. Man versteht darunter die Eigenschaft, auf äußere Ereignisse (z. B. das Anbieten eines Meßwertes) hinreichend schnell (z. B. mit der Ausgabe eines Stellsignals innerhalb einer durch die Prozeßcharakteristik bestimmten Abtastrate) reagieren zu können. Dabei können je nach Prozeß völlig unterschiedliche Reaktionszeiten eine Rolle spielen. Strömungsprozesse etwa (z. B. Trocknungsprozesse) sind langsame Prozesse. Man hat dort relativ viel Zeit zur Informationsverarbeitung. Robotersteuerungen mit optischen Erkennungssystemen hingegen erfordern die Verarbeitung sehr großer Informationsmengen in kurzen Zeitintervallen. Aufschluß über das Reaktionsvermögen eines Prozessors unter Echtzeitbedingungen gibt die Interruptreaktionszeit. Angaben dazu sind aber mit Vorsicht zu werten, da zum Beispiel das Betriebssystem (Interruptbehandlungsprogramme), aber auch die Hardwareumgebung (z. B. Speicherinterface) zu entscheidender Verlangsamung der Interruptverarbeitung führen können.
Der Anwendungsbereich eingebetteter Rechnersysteme läßt sich nach /1/ grob in 3 Teilbereiche einteilen:

- sequentielle Steuerungen (z. B. speicherprogrammierbare Steuerung einer Werkzeugmaschine)
- geschlossene Regelkreise (z. B. Anfahrregelung eines Kernreaktors)
- Datenkommunikation und -verarbeitung (z. B. Bildverarbeitungsteil eines leistungsfähigen Kopiergerätes).

Grundsätzlich kann man für jede dieser Aufgabenklassen auch Universalmikroprozessoren einsetzen. Führende Halbleiterhersteller sind aber insgesamt dazu übergegangen, spezifische Architekturen zu entwickeln, die für bestimmte Anwendungen zugeschnitten sind. Solche Mikroprozessoren heißen im anglo-amerikanischen Sprachbereich Mikrocontroller. Mit dem Mikrocontroller werden wieder jene dominierenden Anwendungsbereiche zurückerobert, für die Mikroprozessoren ursprünglich erdacht wurden, und die sich mit dem Begriff „softwareprogrammierte Steuerung" überschreiben lassen. Zuerst waren Mikroprozessoren für Tischrechner und Terminalsteuerungen vorgesehen. Sie bewährten sich dann auch in der industriellen Steuerungstechnik zur Ablösung „fest verdrahteter" Gerätetechnik. Ab Mitte der 70er Jahre begannen sie, in die von traditionellen Rechenanlagen belegten Marktsegmente einzudringen. Im Vordergrund stand deshalb die Erhöhung der Verarbeitungsleistung (MIPS-Explosion). Zuerst wurden den Minirechnern Marktanteile abgerungen. Mit den PCs eröffneten sich völlig neue Preis-Leistung-Klassen. Gegenwärtig wird sogar der Supercomputerbereich schrittweise erobert. Wenngleich die traditionellen Datenverarbeitungsanwendungen im Vordergrund auch des publizistischen Interesses stehen, so darf nicht übersehen werden, daß weltweit etwa doppelt so viele Mikrocontroller (also spezielle CPUs für eingebettete Computersysteme) verkauft werden wie Universalmikroprozessoren. Bild 1 vermittelt einen Eindruck über die Marktentwicklung /2/. Beherrschung der Mikroprozessortechnik bedeutet also auch und vor allem, den Bereich der eingebetteten Rechneranwendungen zu beherrschen.

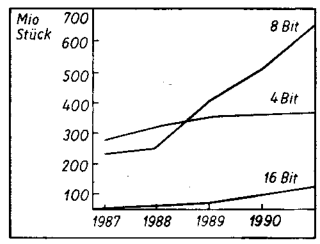


Bild 1 Marktentwicklung bei Mikrocontrollern nach /2/

Im Kombinat „7. Oktober" werden nach eigenen Angaben in Werkzeugmaschinen zur Zeit überwiegend importierte Steuerungen eingesetzt. Damit wird deutlich, daß in der DDR-Volkswirtschaft gerade in diesem entscheidenden Bereich der eingebetteten Rechneranwendungen akuter Nachholebedarf besteht. Auch der Stellenwert der Profillinie „Industrie-PC" ist international offenbar noch nicht ausgestritten. Vor allem in Hinblick auf die zunehmende Nutzung anwendungsspezifischer Schaltkreise (ASICs) erlangen Mikrocontroller überragende Bedeutung. Einseitige strategische Orientierungen mit Überbetonung des Bereiches der Datenverarbeitungstechnik (PCs, Workstations) lassen entscheidende Anwendungsbereiche moderner Mikroelektronik außer acht, die zunehmend für eine moderne Wirtschaft exportentscheidend sind.
Am gesamten Mikrocontrollermarkt hatten Intel-Architekturen 1987 einen Markanteil von 50 %. Auf Grund dieser auch gegenwärtig zu beobachtenden Dominanz sollen im folgenden Intel-Mikrocontroller und -Mikroprozessoren für eingebettete Systeme im Vordergrund stehen. Selbstverständlich bieten aber auch andere Halbleiterhersteller entsprechende Chips an. Intel verfolgt im wesentlichen zwei Architekturkonzepte:

① Ableitung von Spezialarchitekturen aus Universalprozessorlinien (z. B. vom 8086 oder 386) unter Hinzunahme von Peripheriefunktionen mit dem Ziel, einen sehr einfachen Schaltungsaufbau zu unterstützen. Dabei werden Ergänzungsschaltkreise eingespart,

allerdings um den Preis geringerer Leistungsfähigkeit gegenüber der Original-CPU. Es besteht Softwarekompatibilität zum Universalprozessor. Dieser Weg läßt sich begrifflich am besten mit dem Terminus Einchipmikrorechner fassen. Einchipmikrorechner sind besonders für eingebettete Computer, die große Datenmengen verarbeiten müssen (z. B. Bildverarbeitung oder Datennetze), geeignet. Manche Firmen entwickeln Einchipmikrorechner auch mit On-chip-Speichern und auf der Basis einer eigenständigen Architektur ohne Bezüge zu Universalprozessoren.

② Entwicklung spezieller Architekturen ohne Bezug zu Universalprozessorarchitekturen und deshalb auch ohne Softwarekompatibilität. Für diesen Weg soll der Begriff Mikrocontroller verwendet werden. Mikrocontroller verfügen häufig über eigene Speicher, vor allem aber über geeignete, z. T. hoch spezialisierte Interfaces zur Ankopplung an Meß- und Steuerperipherie sowie zur Datenkommunikation (serielle und parallele I/O-Kanäle, Analogein- und Analogausgabe).

On-chip-ROMs oder EPROMs werden zum Teil bei 8- und bei 16-Bit-Systemen angeboten. Die Programme bei 16-Bit- und 32-Bit-Systemen können um Größenordnungen unterschiedliche Längen haben, so daß die Integration von Speicher auf dem Chip meist unökonomisch ist. Statt dessen geht man dazu über, Caches (siehe MP 4/90) und große Registerfiles on-chip zu realisieren, um unzureichende Datenübertragungsraten an der Busschnittstelle auszugleichen.
Einchipmikrorechner und Mikrocontroller können nicht immer eindeutig gegeneinander abgegrenzt werden. Auch in der angloamerikanischen Literatur werden die o. g. Termini nicht einheitlich verwendet. So wird der vom 386 abgeleitete 376 durchaus auch als Mikrocontroller bezeichnet. Bei der Begriffswahl dürften aber auch Werbestrategien eine Rolle spielen...

Einchipmikrorechner – Abkömmlinge von Universalprozessoren

Für die 8-Bit-Welt hat Intel keinen Abkömmling eines Universalprozessors bereitgestellt. Vielmehr wurde mit der Familie MCS-48 gleich der Schritt zum Mikrocontroller vollzogen. Einchipmikrorechner bietet Intel im 16- und im 32-Bit-Bereich an.

Vom 16-Bit-Universalprozessor zum hochintegrierten 16-Bit-Mikroprozessor

Der erste 16-Bit-Prozessor von Intel war der 8086 mit etwa 27000 Transistoren. Durch Verdopplung der Transistorzahl gelang es 1982, im 80186 (kurz 186) zusätzlich zur CPU die wichtigsten Komponenten der Rundumelektronik zu integrieren: Taktgenerator, programmierbarer Interruptcontroller, programmierbarer Timer/Counter (3 × 16 Bit), programmierbarer DMA-Controller (2 unabhängige Kanäle) sowie Bus-Interfaceeinheit und programmierbare Chipauswahllogik für externe Speichereinheiten und Peripherieschaltkreise. Ein programmierbarer Wartezyklengenerator unterstützt die Nutzung langsamer Speicher- und Peripheriebausteine. Der Lokalbuscontroller ermöglicht eine geeignete Externbusschnittstelle für Multibus-Systeme.
Mit dem 186 werden insgesamt 15 bis 20 Einzelschaltkreise funktionell auf einem Chip vereinigt, was eine einfache und billige Leiterplattenrealisierung ermöglicht. Mit der Chipauswahllogik können einzelne Peripherieeinheiten und Speicherblöcke ausgewählt werden. Über programmierbare Konfigurationsregister lassen sich beispielsweise die Speicherbankgröße sowie die Adreßzuordnungen einstellen. Minimalrechnerkonfigurationen kommen mit wenigen Zusatzbausteinen (16-Bit-Adreßlatches und 16-Bit-Datentransceiver) aus. Die Konfigurationsregister für die Controller sind in einem 256 Byte großen Registerblock zusammengefaßt. Bei den Mikrocontrollern MCS-51/96 nennt Intel diese Register einheitlich Spezialfunktionsregister (SFR).
Der 186 ist aus der Sicht des Programmiermodells voll objektcodekompatibel zum 8086. Damit ist der Softwarefundus des 8086 (z. B. aus der Multibus- und aus der IBM-PC-Welt) nutzbar. Einige neue Zusatzbefehle, die später auch im 286 umgesetzt wurden, machen die Programmierung effizienter. Ein entsprechender „Ableger" des 8088, also mit externem 8-Bit-Datenbus bei voller interner 16-Bit-Verarbeitungsbreite, ist der 188.
Den 186 gibt es auch als stromsparende 16-MHz-CMOS-Version 80C186. Die CMOS-Variante des 188 ist der 80C188. Zusätzlich gegenüber den NMOS-Varianten wurde in diesen Schaltkreisen eine DRAM-Refresheinheit integriert (Bild 2). Bemerkenswert sind zwei neue Arbeitsmodi in der CMOS-Variante: ein stromsparender Modus, bei dem durch programmierbare Taktteilung bis herunter auf 500 kHz der Stromverbrauch reduziert wird, und ein spezieller Umgebungstestmodus, der sich in Intel-CMOS-Mikrocontrollern unter der Bezeichnung ONCE eingebürgert hat. ONCE ist von *on circuit emulation* abgeleitet. Durch Hochohmigschalten aller Pins kann der Prozessor von der Hardwareumgebung abgetrennt werden. Der Adapter eines Emulators kann parallel zum Schaltkreis angeklemmt werden, ohne daß der Chip aus der Fassung oder von der Platine entfernt werden muß. Gerade im Gerätebau werden Prüf- und Testverfahren häufig dadurch erschwert, daß die Chips aus Zuverlässigkeitsgründen fest auf der Leiterplatte eingelötet sind. Der ONCE-Modus kommt deshalb der Geräteprüfung in der Fertigung und am Einsatzort besonders entgegen.
Die üblichen Peripherie- und Ergänzungsschaltkreise der 8086-Familie sind auch beim 186/188 verwendbar, zum Beispiel Busarbiter oder Peripheriechips (8251/8255 usw.). Ein Koprozessorinterface (z. B. für die Arithmetik-Koprozessoren 8087 und 80C187) kann mit dem speziellen, integrierten Buscontroller 82188 aufgebaut werden.
Angaben zur Leistungsfähigkeit des 80186 sind in der Literatur nur spärlich enthalten. Die Interruptreaktionszeiten liegen beim 16-MHz-Takt bei einigen 10 µs. Die Verarbeitungsleistung entspricht etwa 50 % der eines 286 (0,5...1 MIPS). Aus Anwendersicht ist für eingebettete Systeme häufig die Kombination eines 80186 mit einem 8-Bit-Mikrocontroller interessant. Der 8-Bit-Mikrocontroller kann beispielsweise „intelligente" I/O-Operationen übernehmen und so den 16-Bit-Prozessor von „langweiliger" und aufwendiger Routinearbeit zugunsten eines höheren Durchsatzes entlasten. An diesem Beispiel wird deutlich, daß Intel keine nicht auch aus Applikationssicht konzeptionell zueinander paßfähigen Prozessor- und Mikrocontrollerlinien entwickelt. Chips unterschiedlicher Verarbeitungsbreiten, Leistungsparameter und Architekturlinien sind miteinander kompensierbar und gestatten für ein breites Aufgabenspektrum den Aufbau von ökonomischer und adäquater Hardware. Besonders wichtig wird die Paßfähigkeitsstrategie im Hinblick auf die Entwicklung anwendungsspezifischer Schaltkreise.

Vom 32-Bit-Universalprozessor zum „Verschnitt" 80376

Zum 286 gibt es keinen entsprechenden „Ableger", wohl aber zum 386 in Form des 80376 (376). Auf einer 130 cm^2 großen Leiterplatte kann mit dieser CPU und dem dazugehörigen Systemperipheriechip 82370 sowie einigen Zusatzschaltkreisen in SMD-Technik ein 32-Bit-Rechner aufgebaut werden, der alle Vorzüge der 32-Bit-Architektur des 386 aufweist, insgesamt aber 30 LSI/VLSI-Schaltkreise ersetzt (32 KByte EPROM, 64 KByte SRAM, 8 DAM-Kanäle, 15 externe Interrupts, 5 16-Bit-Timer, 1 serieller Controller). Ein solcher „eingebetteter Rechner" erzielt bei einer 16-MHz-Taktfrequenz maximal 2,5 bis 3 MIPS. Rechner dieser Leistungsklasse benötigt man zum Beispiel in der Radartechnik, in Da-

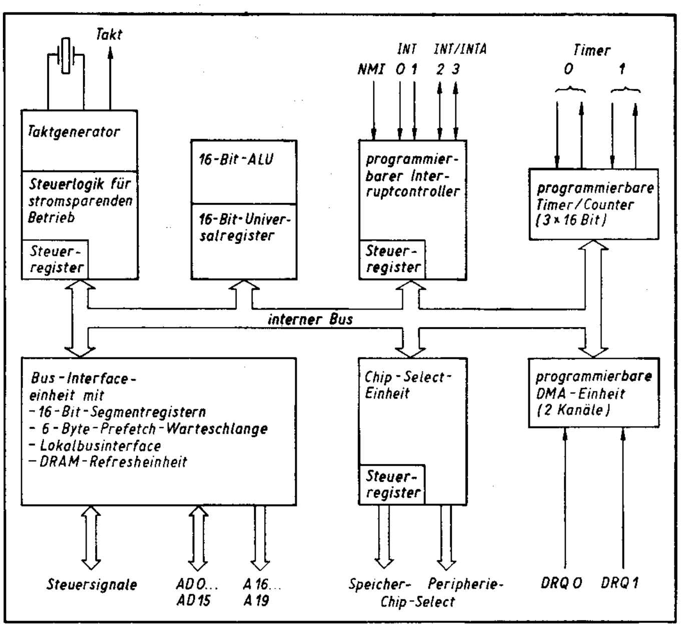


Bild 2
Architektur des 80C186

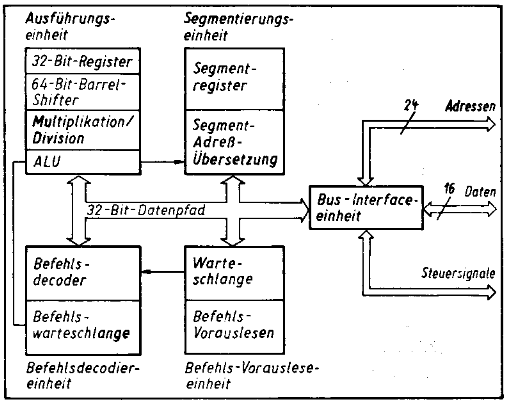


Bild 3 Architektur des 80376

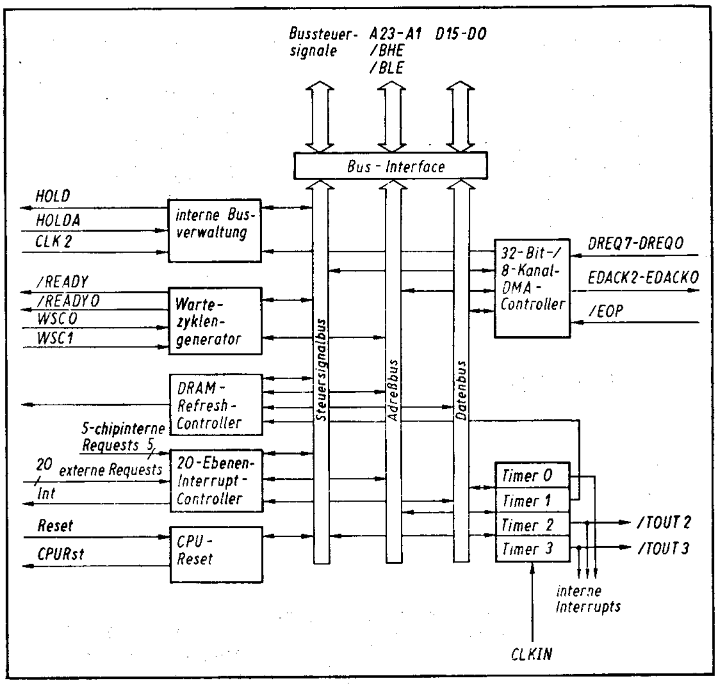


Bild 4 Architektur des 82370

tennetzen (Kommunikationsknotenrechner in Lokalnetzen), in komplexen Robotersteuerungen oder in Laserdruckern für Desktop-Publishing-Systeme. Folgende Architekturunterschiede weist der 376 gegenüber dem 386 auf:
- keine Binärcode-Rückwärtskompatibilität
- Beschränkung auf den *386 protected mode* (Verzicht auf *286 protected mode, real mode* und *virtual 8086 mode)*
- Verzicht auf Seitenverwaltung (paging)
- optimiertes Bus-Interface für 16-Bit-Daten und 24-Bit-Adressen.

Intern ist der 376 wie der 386 voll 32-Bit-organisiert (Bild 3). Eine Prefetcheinheit liest parallel bis zu 16 Byte vom Datenbus. Ein interner Spezialpuffer kann bis zu 3 decodierte Befehle aufnehmen. Damit erreicht der 376 an 16-Bit-Bussen eine höhere Geschwindigkeit als der 386. Die Einschränkung des Adreßraums (16 MByte) ist aus Anwendersicht vertretbar.
Die Auswahl des *protected mode* wurde mit dem Blick auf solche Echtzeitbetriebssysteme wie das iRMK-386 (iRMK-II) getroffen, die Segmentverwaltung und Objektschutz erfordern. Bei Bedarf kann die interne Segmentverwaltung nach erfolgreicher Austestung der Programme abgeschaltet werden.
Das optimierte Businterface des 376 ermöglichte die Reduzierung der Pinzahl. Ein verbessertes Zeitregime unterstützt den Betrieb auch von 100-ns-DRAMs und -EPROMs. Insgesamt erreicht der 376 etwa 70 bis 80 % der Leistungsfähigkeit eines 386 in einem 32-Bit-System.
Als Zusatzschaltkreis gibt es den 82370, einen integrierten Systemperipheriechip (Bild 4). Er enthält die Funktionen DMA-Controller, zum 82C54 kompatible Counter/Interval-Timer, Interruptcontroller, DRAM-Refreshcontroller, programmierbarer Wartezyklengenerator sowie Prozessor-Reset- und Shutdown-Logik für stromsparenden Betrieb. Der 82370 ist eine 16-Bit-Modifikation des 82830. Was beim 80186 also noch in einem Chip untergebracht werden konnte, erfordert nun 2 Chips. Speziell für eigebettete Rechneranwendungen gibt es den 16-K × 16-Bit-EPROM-Speicherschaltkreis 27C203A. Transparente Latches auf der Adreß- und auf der Datenseite verringern den Schaltungsaufwand und gestatten schnelle Datenzugriffe. Die EPROM-Programmierung ist in nur 4 Sekunden möglich und kommt so industriellen Herstellungsprozessen entgegen. Als numerischer Koprozessor steht der 387SX zur Verfügung. Mit interner 80-Bit-Architektur unterstützt er den Standard IEEE 754 und ist objektcodekompatibel zur 8087-Familie. Der 376 ist nicht zu verwechseln mit dem 386SX, der ein Zuschnitt des 386 für 286er Bussysteme ist und den IBM-ATs die 32-Bit-Leistungsfähigkeit des 386 bei geringerem Preis bringt.

Mikrocontroller
Die 8-Bit-Generationen MCS-48 und MCS-51

Intel hat mit der Familie MCS-48 und später mit der Weiterentwicklung MCS-51 ein ganzes Spektrum von architekturkompatiblen 8-Bit-Mikrocontrollern in unterschiedlichsten Konfigurationen geschaffen (verschiedene Speichergrößen, ROM-/EPROM-Varianten, unterschiedliche I/O-Interfaces), die bis in die Gegenwart den Löwenanteil auf dem Controllermarkt innehaben.

MCS-48

Der 8048 wird in HMOS-II-Technologie hergestellt und enthält 17000 Transistoren. Die 8-Bit-ALU hat 8 8-Bit-Register, die zweifach in den RAM agebildet werden (Tafel 1, Bild 5). Der Stack hat 8 Ebenen. 96 Befehle sind verfügbar, davon 90 % Einbytebefehle. Der Befehlssatz hat etwa die Mächtigkeit der 8080-Architektur (Binär-, BCD-Arithmetik), jedoch ohne die 16-Bit-Befehlsgruppen. Die kürzesten Befehlsausführungszeiten liegen bei 1,36 bzw. 2,72 μs. Eine Reihe von Zusatzfunktionen macht diesen Chip zum Mikrocontroller.

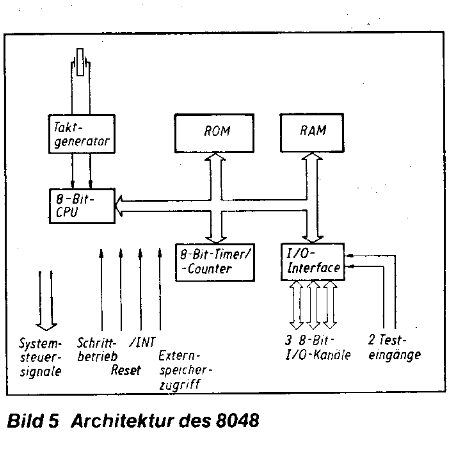


Bild 5 Architektur des 8048

Tafel 1 Konfigurationsvarianten der Familie MCS-48

ROM-Varianten	ROM-loser Typ	EPROM	RAM (Byte)	ROM/EPROM (KByte)
8048AH	8035AHL	8748H	64	1
8049AH	8039AHL	8749H	128	2
8050AH	8040AHL		256	3

Als I/O-Kanäle sind 3 bidirektionale Ports zu je 8 Bit verfügbar. Davon ist eines für ein externes Bussystem nutzbar („8085-BUS-Option"). Mit dem I/O-Expander 8243 kann der I/O-Adreßraum beliebig aufgefächert werden. Auch der Speicher läßt sich über die maximal möglichen 4 KByte ROM und 320 Byte RAM hinaus durch Bankorganisation beliebig erweitern. 2 Biteingänge können vom Programm direkt ausgestattet werden. Als Zusatzfunktionen enthält der 8048 einen voreinstellbaren 8-Bit-Timer/Counter sowie eine Interrupteinheit mit einem Eingang. Der interne Taktgenerator kann im Bereich von 1 bis 11 MHz arbeiten und ist extern synchronisierbar. Damit werden unter anderem die Chiptestung vereinfacht (Synchronisation Tester – Testchip) und der Aufbau von Multimikrocontrollersystemen unterstützt. Der 8048 kann in folgenden Arbeitsmodi betrieben werden: Synchronmodus, Schrittbetrieb, Powerdown und Externbuszugriff. Beim Powerdown-Modus (nur für die ROM-Versionen) bleiben alle RAM-Daten erhalten. Die aufgenommene Leistung kann auf 10 bis 15 % reduziert werden.
Als Zusatzschaltkreis ist unter der 8085-BUS-Option des 8048 die 8080/8085-Peripherie nutzbar. Anwendungen des 8048 sind beispielsweise BCD-/Binärarithmetik, abgesetzte Controller oder Gerätesteuerungen. Einfache digitale Regler (z. B. PID-Regler) sind bei einer 11-MHz-Taktfrequenz mit Abtastraten im ms-Bereich möglich und genügen damit bereits einer Vielzahl praktischer Anforderungen.

UPI-41/42

1977 leitete Intel von der Basisarchitektur des 8048 einen *universal programmable interface slave microcontroller* ab. Dieser Abkömmling ist kompatibel zu den Bussyste-

men der 8- und 16-Bit-Universalprozessoren, aber auch paßfähig zur Familie MCS-51. Slavemikrocontroller werden zum Beispiel in Schrittmotorsteuerungen von Matrixdruckern oder im IBM PC/AT als intelligenter Controller der Multifunktionstastatur eingesetzt. Gegenüber dem 8048 verfügen diese UPIs unter anderem über einen Chip-Select-Eingang, was die Einbindung als Peripheriebaustein in ein Wirtssystem vereinfacht.

MCS-51

Auch der 8051 wurde zunächst in HMOS-II-Technologie gefertigt, und zwar mit etwa 60000 Transistoren. Inzwischen gibt es aber auch CMOS-Varianten (Tafel 2, Bild 6).

Der 8051 enthält eine 8-Bit-ALU mit einem gegenüber dem 8048 erweiterten Befehlssatz. Hervorzuheben sind besonders die Bitverarbeitungsbefehle (Bit transportieren/setzen/löschen/komplementieren, Bits logisch mit UND bzw. ODER verknüpfen, bitabhängige bedingte Programmverzweigungen). Diese Verarbeitungsmöglichkeiten Boolescher Größen (englisch: boolean processing capability) prädestinieren den 8051 für Gerätesteuerungen. Im Intern-RAM sind 128 Bit einzeln adressierbar, aber auch die Bits aller Spezialfunktionsregister (SFR) zur Steuerung des Datenaustauschs mit der Umgebung kann man einzeln erreichen.

Neu sind 8 × 8-Bit-Multiplikations- und Divisionsbefehle.

Die Adreßräume umfassen je 64 KByte für Daten und Programme. Der 8051 enthält 2 16-Bit-Timer/Counter. 4 bidirektionale I/O-Ports sind universell nutzbar, zum Beispiel im Zusammenwirken mit den Timern für eine einfache Analogausgabe mit Pulsbreitenmodulation oder auch zur Gestaltung eines externen Bussystems. 5 Interruptquellen werden in 2 Privilegierungsebenen ausgewertet. Der Taktgenerator ist wieder on-chip (12 MHz, extern synchronisierbar).

Die Hardwareressourcen sowie die Prozessorarbeitsmodi können über Spezialfunktionsregister (SFR) durch die Software programmiert werden. Die SFR werden in einem Registerfeld adressiert, das je nach Spezialkonfiguration des Prozessors (z. B. unterschiedliche Anzahl von Timern) entsprechend strukturiert ist.

Die meisten Befehle brauchen 12 Takte zu ihrer Abarbeitung, Multiplikation und Division 48 Takte, einige Befehle erfordern 24 Takte. Bitbefehle werden etwa 5- bis 15mal schneller abgearbeitet als entsprechende Befehlsfolgen auf dem 8048. Als Mittelwert lassen sich mehr als 0,5 Millionen Befehle je Sekunde angeben (8-Bit-orientiert). Der 8051 ist damit etwas schneller als der 8048, aus Anwendersicht aber wesentlich breiter und flexibler einsetzbar. Die CMOS-Varianten können in zwei stromsparenden Arbeitsmodi (Anhalten der CPU) betrieben werden. Im Idle-Modus bleiben die Controller aktiv und die RAM-Inhalte gesichert. Bei Power-down werden nur die RAM-Daten gesichert. Bei der CMOS-EPROM-Variante 87C51 gibt es den beim 80C186 bereits vorgestellten ONCE-Modus zur Unterstützung der On-circuit-Emulation.

Als Zusatzschaltkreise sind die 8080/8085-Peripheriecontroller einsetzbar.

Für Speichererweiterungen gibt es Spezial-EPROMs in der Organisation 8 K × 8. Der 87C64 ist zum Anschluß an im Multiplex betriebene Daten- und Adreßbusse ausgelegt. Der 27C64 ist für getrennte Bussysteme für Daten und Adressen vorgesehen. Beide Varianten enthalten bereits Zusatzlogik zur Vereinfachung des Schaltungsaufbaus (Latches und Demultiplexer). Der EPROM 87C257 hat eine 32-K × 8-Organisation. Hervorzuheben sind bei den MCS-51-Varianten mit on-chip-EPROM bzw. -ROM Maßnahmen zum Ausschluß unberechtigten Zugriffs (Softwareschutz). Mit einer 32 Byte langen Codierungsinformation können die ROM/EPROM-Inhalte exklusiv und ODER-verknüpft werden, so daß die Bedeutung der Bytes nur bei Kenntnis der Codierungsinformation sichtbar wird. Außerdem kann man Lesezugriffe auf den Internspeicher unterbinden, die Nachprogrammierung des EPROMs verhindern und (als härteste Stufe des Zugriffsschutzes) die Prüfung des EPROM-/ROM-Inhalts verbieten. Die EPROM-Varianten sind nach dem Intel-Quick-Pulse-Verfahren programmierbar (8 KByte in 25 Sekunden). Dabei wird die Programmierung der EPROM-Zellen beendet, sobald die Information stabil eingetragen ist.

Für externe RAM-Erweiterungen gibt es den iRAM 51C86. Das ist ein DRAM-Schaltkreis mit der Organisation 8 K × 8, der aber über einen internen Refreshcontroller verfügt und deshalb nach außen wie ein statischer RAM nutzbar ist. Intel bezeichnet diese RAMs als pseudostatische RAMs.

Der 8051 wird vor allem für Kommunikationsprobleme eingesetzt, zum Beispiel zur Kommunikation nach CSMA/CD (Ethernet), HDLC oder SDLC mit bis zu 2 MBaud. Weitere Anwendungen sind beispielsweise intelligente Controller in Rechnern (z. B. Tastaturcontroller oder Ansteuerung eines LC-Displays 25 × 80 Zeichen) und die Lösung von Permutationsaufgaben (z. B. Codierungsaufgaben in Bankterminals nach dem im USA-Bankwesen verbindlichen Data Encryption Standard).

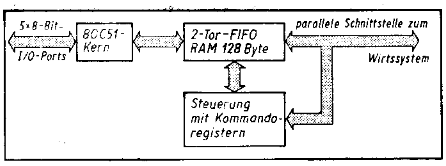


Bild 7 Architektur des UPI-452

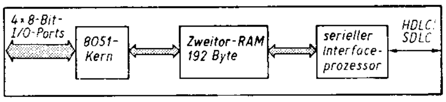


Bild 8 Architektur des RUPI-44

Die Familie MCS-51 kann als Plattform für eine Vielzahl von Spezialcontrollern angesehen werden, die für unterschiedliche Aufgabenstellungen zurechtgeschneidert wurden. Drei Vertreter seien im folgenden kurz vorgestellt.

UPI-452

UPI-452 ist ein universeller, frei programmierbarer Slave-Mikrocontroller in CMOS-Technologie auf der Basis des MCS-51 (Bild 7). Er enthält ein 128 Byte tiefes, bidirektionales FIFO-Slaveinterface. Dieser FIFO ist ein Zweitorspeicher mit der Zugriffsstrategie *first in – first out*. Er dient dem Informationsaustausch des Mikrocontrollers mit der Umwelt. Dieser FIFO-Speicher kann beliebig auf Eingabe- und Ausgabeseite aufgeteilt werden. Der 2-Kanal-DMA-Prozessor gestattet den Transfer von 64-KByte-Blöcken. Mit UPI-452 wird das Problem des Datenaustauschs zwischen schnellen 286er Systemen und langsamerer Peripherie effektiv gelöst.

RUPI-44

Der Name RUPI ist abgleitet von *remote universal programmable interface slave microcontroller*. Das heißt soviel wie universell programmierbarer Interface-Slave-Mikrocontroller für abgesetzte Systeme. Die RUPI-44-Familie (Bild 8) enthält Einchipmikrocontroller auf der Basis des MCS-51, erweitert um einen Controller für ein serielles Hochleistungsinterface für das HDLC/SDLC-Protokoll, das Daten mit 2,4 MBaud bei Fremdtaktung und 375 kBaud bei Interntaktung übertragen kann. Der serielle Teil kann unabhängig vom MCS-51-Kern arbeiten. Interfaceeinheit und MCS-51-Kern kommunizieren über einen internen 192-Byte-Dual-Port-RAM.

RUPI-Prozessoren sind für den Einsatz in Hochgeschwindigkeitsnetzen oder in schnellen verteilten Systemen vorgesehen und erfüllen dort komplexe Steuerungs- und Datenaustauschaufgaben. Ein Anwendungsbei-

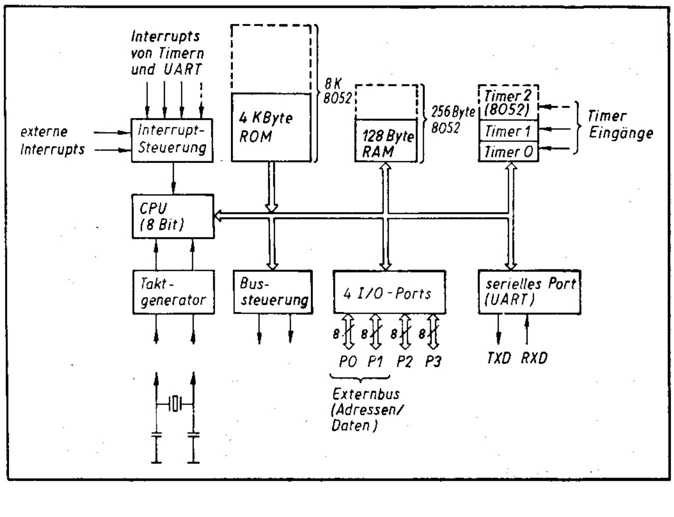


Bild 6 Architektur des 8051

Tafel 2 Konfigurationsvarianten der Familie MCS-51

Typ	ROM-loser Typ	EPROM-Typ	ROM (KByte)	RAM (Byte)	16-Bit-Timer	Technologie
8051	8031	(8751)	4	128	2	HMOS
8051AH	8031AH	8751H	4	128	2	HMOS
8052AH	8032AH	8752BH	8	256	3	HMOS
80C51BH	80C31BH	87C51	4	128	2	CMOS
83C51FA	80C51FA	87C51FA	8	256	4	CMOS
83C152	80C152	87C152	8	256	2	CMOS

spiel sind untereinander vernetzte, intelligente Sensoren in einem räumlich verteilten Meßwerterfassungssystem (Prozeßautomatisierung).

80C152
Als universelle Kommunikationscontroller sind die mit 16,5 MHz getakteten Mikrocontroller 80C152 konfiguriert. Neben dem Kerncontroller MCS-51 enthalten diese Chips unter anderem einen Mulitcontroller-Kommunikationscontroller (SDLC, HDLC, CSMA/CD oder nutzerdefiniertes Protokoll, bis 2,4 MBaud) für Voll- oder Halbduplexbetrieb und einen 2-Kanal-DMA-Controller. Besonders auf kostenempfindliche Anwendungen im Hochgeschwindigkeitskommunikationsbereich einschließlich Lokalnetzanwendungen und serielle Rückverdrahtungsbusse im Gerätebau zielt diese Architektur. Auch von diesen Chips gibt es wieder unterschiedliche Speicherkonfigurationen (RAM/ROM/EPROM).

Zusammenfassung
Intel hat mit der Familie MCS-51 eine universelle Basisarchitektur entwickelt, die für die Mehrzahl der gegenwärtig und künftig zu lösenden Aufgaben ökonomisch einsetzbar ist. Im letzten Abschnitt wird skizziert, wie dieses Architekturkonzept im ASIC-Entwurf umgesetzt wird. Dieser De-facto-Industriestandard dürfte uns deshalb wohl auch in den kommenden Jahren begleiten, denn noch immer gibt es Weiterentwicklungen. Erst 1989 wurde zum Beispiel der 83C51GB vorgestellt, der ein programmierbares Timer-/Counterfeld enthält und für Anwendungen im Automobilbau bestimmt ist (z. B. Zündungssteuerung, Antiblockiersysteme, Sitzverstellung, Klimatisierung, Bordcomputer, Einspritzsteuerung, elektronische Fensterverstellung mit Sicherheitsfunktion usw.) /7/.
MCS-51-Zweitlieferanten sind in Westeuropa zum Beispiel Siemens und Valvo/Philips. Diese Hersteller haben zum Teil auch eigene Modifikationen entwickelt.

Die 16-Bit-Generation

Für Steuerungsanwendungen mit Analogwerteingabe und -ausgabe sowie mit höheren Anforderungen an die Verarbeitungsleistung wurde die 16-Bit-Mikrocontrollerfamilie MCS-96 mit einem Integrationsgrad von 120000 MOS-Gattern (ROM-Version) entwickelt. Neben der NMOS-Variante 8096BH und davon abgeleiteten Konfigurationen (1. Generation) gibt es auch CMOS-Varianten (2. Generation) mit entsprechend verbesserten Leistungsparametern und vervollkommneter Architektur (Bild 9). Ein 256-Byte-RAM enthält unter anderem die bereits aus der 8-Bit-Generation bekannten konfigurationsabhängig belegten Spezialfunktionsregister SFR, Datenrettungszellen sowie ein Universalregisterfile in enger Kopplung mit der 16-Bit-ALU. ROM-/EPROM-Varianten enthalten 8 KByte Speicher auf dem Chip. Ein A/D-Umsetzer (CMOS-Variante: 26 bzw. 49 μs Umsetzzeit bei 10-Bit-Auflösung und 12-MHz-Takt) mit 4- oder 8-Kanal-Multiplexer und analogen Haltegliedern gestattet die Eingabe analoger Signale. Eine spezielle mehrkanalige Hochgeschwindigkeitseinheit zur Eingabe und zur Ausgabe digitaler Signale, verschiedene Multifunktions-I/O-Ports, mehrere Timer mit Captureregistern (das sind Register zur Erfassung von Zählzwischenständen), ein pulsbreitenmodulierter Analogausgang (bei 12-MHz-Takt Signalfequenz 23,6 kHz) sowie zur Funktionsüberwachung ein integrierter Watchdogtimer. Der Adreßraum umfaßt 64 KByte. Die Busschnittstelle kann wahlweise im MCS-51-Modus oder im 16-Bit-Adreß-/Datenmultiplex betrieben werden. 21 Interruptquellen werden auf 8 Interruptebenen gelenkt, davon 1 externer Interrupt. Die CMOS-Variante 80C196KA ermöglicht Multiplikationen von 16 × 16 Bit in 2,33 μs, die Interruptreaktionszeit beträgt 0,33 bis 6,5 μs. Die Befehlsausführungszeiten liegen für die NMOS-Varianten zwischen 1 und 3 μs, für die CMOS-Varianten bei 0,5 bis 1,5 μs.
MCS-96-Chips verfügen (je nach Variante) über einen Eingang und einen Ausgang zur externen Taktsynchronisation. Außerdem ist bei den CMOS-Varianten wieder der ONCE-Modus für die On-circuit-Emulation nutzbar. Die aus dem MCS-51 bekannte ROM-/EPROM-Zugriffssperre blockiert unberechtigte Zugriffe auf ROM-/EPROM-Inhalte.
Zur schnellen EPROM-Programmierung wird der modifizierte Quick-pulse-Algorithmus verwendet. MCS-96-Chips unterstützen 3 Programmiermodi: Selbstprogrammierung, Slaveprogrammierung und Programmierung während des Programmlaufs. Grundsätzlich genügt es, eine entsprechende Spannungs- und Signalbelegung zu aktivieren, um den Programmiervorgang auszulösen. Bis zu 15 Slaves können über ein gemeinsames Bussystem von einem Master aus programmiert werden. Eine Nachprogrammierung während des Laufs wird zum Beispiel beim nachträglichen Einprägen von Parametern benötigt. Beim „Einfahren" von Adaptivreglern kann man zum Beispiel einfache Prozeßidentifikationsalgorithmen (Parameterschätzverfahren) realisieren, die vor Ort die notwendigen Reglereinstellparameter ermitteln und einen fest programmierten Regelungsalgorithmus damit „füttern". Eine solche Inbetriebnahme vor Ort bedeutet höhere Flexibilität in der Anwendung eines Adaptivreglers. Ein interessantes Applikationsbeispiel für diesen Prozessor ist ein Antiblockiersystem für PKWs /10/. Über 4 Geschwindigkeitssensoren werden die Drehzahlen jedes Rades aufgenommen. Zur Dateneingabe werden die Hochgeschwindigkeitseingänge genutzt. Die CPU berechnet daraus die Umdrehungsgeschwindigkeiten und die Bremsverzögerungen. Die Ergebnisparameter werden an die Bremsregler übermittelt, die die Bremskraft jedes Rades steuern. Der gesamte Regelvorgang darf höchstens 5 ms dauern.
Eine andere Anwendung ist die in /9/ vorgestellte Heizungssteuerung für Warmwasserheizung mit Sonnenkollektor. Ein 16-Bit-Mikrocontroller steuert den Verbrennungsprozeß einer Warmwasserheizung entsprechend den Temperaturwerten in Heizungsvorlauf und -rücklauf. Der Anwender kann unter anderem Tagesgänge des Temperatursollwertes programmieren. Außerdem wertet die Steuerung die von einem Sonnenkollektor aufgenommene Energiemenge aus und lenkt den Wasserstrom gegebenenfalls um.
Für die Basisversion 8096 wird in /1/ die Software für eine schnelle Fouriertransformation mit reellen Eingabewerten vorgestellt. Bei einer Verarbeitungsbreite von 16 Bit braucht der komplexe Basis-8-FFT-Algorithmus bei 64 Punkten 12,5 ms (zuzüglich 6,4 ms zur Analogwerterfassung bei einer Abtastfrequenz von 156 Hz) und bei 1024 Punkten 300 ms. Zum Vergleich: Moderne 16- und 32-Bit-Signalprozessoren benötigen bei 64 Punkten Rechenzeiten von einigen 100 μs bis einige ms.

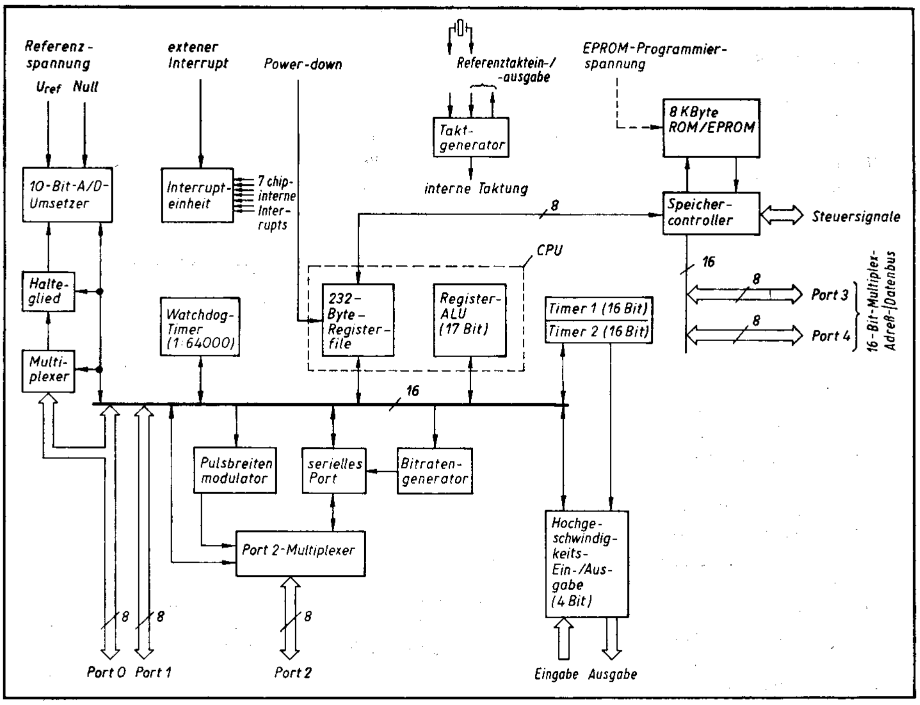


***Bild 9** Architektur des 80C196 KA*

Die 32-Bit-Generation 80960

1988 wurden erste Veröffentlichungen zum 80960 bekannt. Mit diesem Chip eröffnete Intel seine 32-Bit-Mikrocontrollerlinien, für die es gegenwärtig bereits mehrere kompatible Varianten in zwei Generationen gibt.

1. Generation: 80960 K bzw. Mx (x: Konfigurationskennzeichnung), bei 25-MHz-Taktfrequenz bis zu 9 VAX-MIPS/1,2 MFLOPS

• 80960 KA – 32-Bit-Mikrocontroller mit RISC-Kernprozessor und Interruptcontroller
• 80960 KB – wie 80960 KA, zusätzlich On-chip-Gleitkommaeinheit
• 80960 MB – Mikrocontroller für militärische Anwendungen mit hohen Leistungs- und Sicherheitsanforderungen, Unterstützung von Speicherverwaltung und Multitasking sowie Fehlertoleranz

2. *Generation:* 80960 XA und CA – erweiterte Architektur mit verbesserter Betriebssystemunterstützung und wesentlich gesteigerter Verarbeitungsleistung bis maximal 66 MIPS.
Die 32-Bit-Mikrocontroller folgen einem sehr interessanten und an Details reichen Architekturkonzept, deshalb stellen wir diese Prozessoren in einem separaten Beitrag in MP 6/90 vor.
Die Architekturen von 32-Bit-Mikrocontrollern basieren auf RISC-Gestaltungsprinzipien. Trendeinschätzungen besagen, daß der 32-Bit-RISC-Markt vor allem im Mikrocontrollerbereich mit 300 % jährlich verhältnismäßig schneller wachsen wird als der CISC-Markt, ganz besonders im Bereich der Produktionssteuerung. Für Robotersysteme werden in den kommenden Jahren besonders hohe Zuwachsraten erwartet /15/.
Nach Angaben von LSI-Logic sind ASICs mit 32-Bit-RISC-Makrozellen zu einem Preis von 10 $ je MIPS möglich /3/. CISC-Architekturen kosten rund 100 $ je MIPS.

Entwicklungsunterstützung für den Hard-/Softwareentwurf

Für alle Mikrocontroller gibt es ein adäquates Spektrum an Hard- und Softwareentwurfshilfsmitteln. Dazu zählen:
– leistungsfähige Makroassembler
– Compiler für PL/M (die Haus-Systemprogrammiersprache von Intel) und C
– Echtzeitbetriebssysteme bzw. Betriebssystemkerne
– Programmierumgebung (modulare Programmierung, Bibliotheksverwaltung, Programmpakete – z. B. Gleitkommabibliotheken)
– Hard- und Softwaretestunterstützung (In-circuit-Emulatoren, Echtzeitdebugger)

Die Softwareentwicklung folgt der von den Universalprozessoren her bekannten Technologie (Übersetzen, Binden, Relokalisieren, Bibliotheksverwaltung usw.). Softwaretools sind mit einheitlichen Nutzerinterfaces ausgestattet. Dadurch wird der Umstieg auf eine neue CPU oder einen neuen Mikrocontroller erleichtert.
Der Komplexentwurf und die Verbundtestung von Hardware und Software wird auf der Grundlage von leistungsfähigen Emulatoren (z. B. ICE-51) und Echtzeitdebuggern (z. B. PSCOPE-51) unterstützt. Entwicklungsumgebungen sind auf den traditionellen Intel-Entwicklungssystemen (INTELLEC-III und IV), auf IBM-PCs oder zum Teil auch auf Minirechnern (VAX und MICROVAX) einsetzbar. Die Softwaretools laufen unter verschiedenen Betriebssystemen, zum Beispiel unter RMX-Systemen, unter UNIX, MS-DOS oder VMS. Diese Konsistenz und Kompatibilität der Entwickungshilfsmittel wird auch bei der im folgenden Abschnitt beschriebenen Entwurfsbasis für anwendungsspezifische Schaltkreise bewahrt.

Tafel 3 Elemente der CHMOS-III-Intel-MCS-51-Zellenbibliothek

Typ	Funktion
UCS51nnX	Prozessorkern 8051 (nn × 4 KByte ROM, x = A: 128 Byte RAM, 5 Interruptquellen; x = B: 128 Byte RAM, 10 Interruptquellen; x = C bzw. D: wie A bzw. B, aber 256 Byte RAM)
UCS51BIU	Bus-Interface-Unit zur Anbindung von Anwenderlogik an den Spezialfunktionsregisterblock
UCS51AD	8-Bit-Umsetzer (sukzessive Approximation, Umsetzzeit 20 μs bei 16 MHz)
UCS51T2	16-Bit-Timer/-Counter mit 3 Arbeitsmodi
USC51SIO	serielle E/A-Schnittstelle (vollduplex, 5 Betriebsarten, variable Bitrate, automatische Paritätsgenerierung)
UCS51BRG	programmierbarer Bitratengenerator (50 Hz ... 19,2 kHz)

Mikrocontroller und ASICs

Anwendungsspezifische Schaltkreise (ASICs) können international gegenwärtig mit einem Integrationsgrad von bis zu 10^6 Transistorfunktionen gefertigt werden. Damit ist es möglich, einen Mikroprozessor oder einen Mikrocontroller als Kernprozessor (englisch: core processor) mit der gesamten anwendungsspezifischen Umgebungselektronik auf einem Chip zu integrieren. International wurden Komplexitäten des Kernprozessors bis zur Größenordnung eines 286 bekannt. Der Leiterplattenentwurf wandelt sich zum ASIC-Entwurf.
Bemerkenswert ist die ASIC-Marktstrategie von Intel. Nach einem nicht sehr glücklichen Versuch im Gate-Array-Bereich entschloß man sich, die als Industriestandards bewährten Prozessorlinien einschließlich der Ergänzungsschaltkreise in ASIC-Bibliotheken anzubieten. Damit treten folgende Vermarktungseffekte ein:
– Sicherung des De-facto-Industriestandards bei Prozessorlinien auch für den ASIC-Bereich (Wahrung der Kompatibilität von Hardware und Software auf Rechner-Systemebene)
– schritthaltende Umsetzung bewährter Architekturlinien in Form von neuen technologischen Verfahren.
Tafel 3 vermittelt einen Eindruck vom Angebot an MCS-51-Funktionen in der 1,5-μm-CHMOS-III-Zellenbibliothek von Intel. Die Intel-Entwurfssoftware enthält die Funktionen Schaltplaneingabe, Timingsimulation (Gatterebene), Programmentwicklung, Systemtest, Generierung der Testpattern (für die UCS-51-Zellen gibt Intel die Testvektoren vor) und Entwurfsverifikation. ICE-UCS51 ist ein ASIC-in-circuit-Emulator zur Komplextestung von Hardware und Software in der Zielhardware. Neben den UCS-51-Kernen kann auch die Anwenderlogik emuliert werden.
Die Entwurfssoftware läuft auf Mentor-Grafik-Arbeitsstationen, die Referenzsimulation auf VAX-Rechnern mit Hardwarebeschleunigern. ASICs auf der Basis des UCS-51 können bis zu etwa 10000 Gatter Zusatzlogik enthalten. Für Emulationszwecke gibt es eine spezielle Bondvariante des Prozessorkerns. Die anwendungsspezifische Logik kann mit EPLDs (löschbare programmierbare Logikbauelemente) nachgebildet werden. Noch vor einer Musterpräparation kann man so die Systemeigenschaften an einem in diskreter Technik aufgebauten Modell untersuchen. Zur Programmentwicklung kann der erweiterte Assembler ASM-51, der auch die Testpattern generiert, genutzt werden.

Als nächstes plant Intel, auch 16-Bit-Prozessorkerne dem ASIC-Entwickler zu erschließen. Grundlage ist jeweils das Original-Layout des CMOS-Prozessors.
Auch andere führende Halbleiterhersteller (z. B NEC oder Motorola) bringen in die Zellenbibliotheken der ASIC-Entwurfssysteme bewährte Prozessoren als Kerne ein. Umgekehrt nehmen die Anbieter von ASIC-Entwurfssystemen standardisierte Kernprozessoren mit Ergänzungsschaltkreisen in die Zellenbibliotheken auf. So findet man neben den Mikrocontrollern gegenwärtig selbst Universalprozessoren einschließlich der Unterstützungsschaltkreise (z. B. 8080-Interfaceschaltkreise wie 8251, 8255, 8257 usw.) in den Bibliotheken wieder. Auch die Anbieter moderner RISC-Prozessoren erschließen sich zusätzliche Vermarktungsmöglichkeiten über den ASIC-Bereich (z. B. Motorola 88000, Sun SPARC, MIPS R3000).

Ausblick

Mit Mikrocontrollern und eingebetteten Mikroprozessoren im Gesamtspektrum der gegenwärtig üblichen Verarbeitungsbreiten und Verarbeitungsleistungen wird eine Vielzahl von zum Teil noch nicht erschlossenen Marktsegmenten überdeckt. Der Trend zur Nutzung anwendungsspezifischer Schaltkreise führt dazu, daß bewährte Architekturkonzepte übernommen werden und Kernstück der Zellenbibliotheken werden. Ein Geräteentwurf ohne ASICs ist künftig nicht mehr denkbar. Die Wiederverwendung von bewährten Mikrocontroller- und Mikroprozessorlinien ist ein Gebot der Akzeptanz und der Ökonomie. Halbleiterhersteller sichern die Vermarktung von Architektur-Know-how zusätzlich über die ASIC-Strecke ab, indem sie bewährte Konzpete als Zellen einbringen. Daraus ergibt sich die Forderung an den Schaltkreisentwurf, Architektur und Schnittstellen neuer Prozessor- und Mikrocontrollerfamilien von Anfang an so zu gestalten, daß später eine Übernahme in Zellenbibliotheken problemlos möglich ist.

Literatur

/1/ Intel: Embedded Controller Handbook. Vol. I & II. 1988
/2/ Intel mit guten Karten. Elektronik, München 36 (1987) 21, S. 24
/3/ Scharf, A.: ASICs sichern Know-how. Hard and Soft (1989) 1, S. 77
/4/ Intel: Microprocessor and Peripheral Handbook, Vol. II – Peripheral. 1988
/5/ Förster, J.: Diskrete Fouriertransformation und digitale Signalverarbeitung. Mikroprozessortechnik, Berlin 3 (1989) 8, S. 243
/6/ Kroll, M.: ASIC-Entwicklung auf der Basis des 80C51. elektronik industrie, München 20 (1989) 9, S. 65
/7/ Preiß, E.: 8-Bit-Mikrocontroller mit vielen Möglichkeiten. elektronik industrie, München 20 (1989) 9, S. 98
/8/ Baum, E.: Mehr UARTs für Ihre „51er"! elektronik industrie, München 20 (1989) 2, S. 78
/9/ Bernstein, H.: Heizungsregelung mit 16-Bit-Mikrocontroller. Design & Elektronik, Haar/München 5 (1989) 10, S. 72
/10/ McIntyre, S. M.: Ein 16-Bit-Controller als Turbo für ABS-System. Design & Elektronik, Haar/München 5 (1989) 18, S. 73
/11/ Bodenkamp, J.: „Superscalar-Technologie": Die zweite Generation der 80960-Architektur. Design & Elektronik, Haar/München 5 (1989) 21, S. 6; 22, S. 6; 23, S. 6
/12/ Purkiser, C.; Kardach, J.: The Intel 376 Family for Embedded Processor Applications. IEEE Micro 8 (1988) 3, S. 10
/13/ Bride, P.; Baker, T.: 80960 s RISC-Like Architecture. VLSI System Design (1988) 6, S. 18
/14/ Ryan, D. P.: Intel 's 80960: An Architecture Optimized for Embedded Control. IEEE Micro 8 (1988) 3, S. 63
/15/ New CPUs spearhead Intel attack on embedded computing. Computer Design 27 (1988) April 15, S. 22

Fraktale als Interferenzeffekte

Gabriele Buhren, Berlin

Zeitabhängig komplexe Zahlen in Polarkoordinatenform kombinieren über Betrag und Phase die beiden bekannten Bewegungsarten *geradlinig* und *kreisförmig*. Gleichzeitig werden auf diese Weise ebene Wellen[1] beschrieben, die unter bestimmten Bedingungen miteinander interferieren.

Bezüglich der belebten Materie wird hier die Hypothese geäußert und begründet, daß rückgekoppelte Schwingungen, hervorgerufen durch relativ konstant schwingende Molekülketten, für die Existenz aller Organismen von grundlegender Bedeutung sind.

Springende Spiralen

Bildet man das Quadrat einer komplexen Zahl $Z = r\ \exp(ip)$, wird r quadriert und die Phase $p = -wt$ verdoppelt. Man kann auch sagen, w bleibt unverändert, und t würde verdoppelt, so daß die Funktion z^k dann k Zeitschritten entspricht, in denen der komplexe Zeiger jeweils um w weiterrückt. Für $r = 1$ ist durch Z^k die ungestörte Rotation des Zeigers für k Zeitschritte zu verstehen, ansonsten wird die Zeigerspitze eine steile Spirale nach innen ($r < 1$) oder außen ($r > 1$) ausführen, nach Null oder Unendlich.

Um diese spiralförmige Implosion oder Explosion aufzuhalten und in ein stationäres Verhalten zu bringen, müßte ab und zu additiv in Richtung Einheitskreis eine Gegenkraft wirken. Ein Zentralfeld könnte diese Aufgabe vorzüglich erledigen, soll aber hier nicht betrachtet werden, sondern ein an allen Punkten paralleles Vektorfeld $C = r\exp(ip)$, weil es zur Apfelmännchen-Familie

$$Z = z^k + C, \quad z = Z \tag{1}$$

führt. Für k gegen Unendlich bildet (1) den Einheitskreis ab (siehe Bilder 2 und 3 für $k = 3$ und $k = 7$; alle Bilder auf der 4. Umschlagseite).

Multipliziert man verschiedene komplexe Zahlen, wird pro Zeitschritt w geändert und r auch ungleichförmig bewegt. Die Addition des Vektors C aber erzeugt im günstigen Fall einen Gegensprung in r und p, das heißt eine Kompensation der durch die Multiplikationen erzeugten Beschleunigungen. Soll in (1) die Gegenkraft häufig wirken, dann addiert man C nach jedem Zeitschritt (= Apfelmännchen); soll sie im Vergleich zur Explosion schwächer sein (mit kleinerer Frequenz), dann erst nach mehreren Potenzen von Z, das heißt mit $k > 2$.

Betrachten wir die Beispielgleichung $Z = z^5 + C$. Beim konkreten Punkt C habe sich folgendes Verhalten herausgebildet:

$$\begin{aligned} Z1 &= z3 \times z3 \times z3 \times z3 \times z3 + C \\ Z2 &= z1 \times z1 \times z1 \times z1 \times z1 + C \\ Z3 &= z2 \times z2 \times z2 \times z2 \times z2 + C \end{aligned}$$

Nach jeweils vier Zeitschritten (Multiplikationen) wirkt einmal die Gegenkraft C. Wenn sie dreimal gewirkt hat, ist ein ehemaliges Z wieder erreicht. Der Vorgang $Z1 \rightarrow Z2 \rightarrow Z3 \rightarrow Z1$ entspricht einer Schwingung, die sich auch durch

$$Z3 = ((z3^5 + C)^5 + C)^5 + C$$

beschreiben läßt. Dies ähnelt sehr einer Schwebung.

Die Zeigerspirale kann durch ein konkretes C noch schneller auf einen Punkt stürzen oder nach Unendlich springen (Divergenz). Sie kann aber auch in regelmäßige (periodische) oder unregelmäßige (chaotische) Pulsation geraten.

Komplexe Wellendarstellung und Interferenz

Im Wellenmodell stellt sich das Ganze als Interferenz von immer mehr Wellen dar. Es genügt die Eulersche Winkeldarstellung, um alle Punkte des komplexen Fraktals (hier stellvertretend Apfelmännchen) als stabile Schwebungszustände interpretieren zu können. Die „Urwelle" sei

$$C = r\exp(ip) \quad \text{mit}\ p = kx - wt \tag{2}$$

Man braucht nur die Iterationsfolge analytisch auszumultiplizieren und kommt für die ersten Werte auf

$$\begin{aligned} Z1 &= C = r\exp(ip) \\ Z2 &= C + Z1 \times Z1 = r\exp(ip) + r^2\exp(i2p) \\ Z3 &= C + Z2 \times Z2 = r\exp(ip) + r^2\exp(i2p) \\ &\quad + 2r^3\exp(i3p) + r^4\exp(i4p) \end{aligned}$$

Bei jeder Multiplikation entstehen neue Harmonische. Die Summanden haben die Form $S(m) = A_m r^m \exp(imp)$. Für Winkel mp, die sich gegenüberliegen oder am Kreis symmetrisch angeordnet sind, lassen sich immer r-Werte finden, die zur vektoriellen Teilsumme Null führen. Diese Frequenzanteile heben sich dann gegenseitig auf, sind gelöscht. So kann es auch vorkommen, daß ein früherer Zustand exakt wieder erreicht wird. Der Vorgang wiederholt sich, das heißt, es erfolgt eine Reduktion von Z(n) auf Z(n – j), und Z durchläuft immer wieder denselben j-Zyklus.

Beispiel:
Summanden 3 und 4 in Z3 für $r = 2$, $p = pi$:
$$-2r^3 + r^4 = 0.$$

Wird durch die Auslöschung kein früherer Zustand erreicht, aber der Bereich des Auslöschen-könnens nie verlassen, liegt stabiles chaotisches Verhalten vor, zum Beispiel auf den dünnen Antennenlinien.

Zum Fraktal gehören nur Punkte, die durch Auslöschung erfolgreich das divergente Explodieren verhindern. Wenn wir auch unter Divergenz das Abtrennen von Molekülen aus einer Gruppe verstehen, dann muß sich diesem Auslöschungsprinzip analog der Aufbau von Makromolekülen unterordnen.

Ketten aus Variablen, Vierpolen und DNS-Basensequenzen

Es sind auch gemischte Ketten von Z-Potenzen denkbar, wie

$$Z = z^3 - z^5 + C \qquad \text{(Bild 5)} \tag{3}$$

oder die Mischung verschiedener C-Ebenen:

$$Z = (z^4 + C1)^3 + C2.$$

Darunter fällt das Konzept der verkoppelten Zwillinge /3/:

$$\begin{aligned} Z1 &= z1^2 + C1\,(z2) \\ Z2 &= z2^2 + C2\,(z1) \end{aligned} \tag{4}$$

die zwar parallel zueinander iteriert werden, aber über eine Kopplung in Verbindung stehen. Mit den beiden Gleichungen

$$C1 = C \times (1 + a \times z2),\ C2 = C \times (1 - a \times z1) \tag{5}$$

beobachten wir eine Metamorphose des Apfelmännchens. Für ein festes a (Evolutionsfaktor) wird jeweils ein Bild erzeugt. Mit wachsendem a schiebt das Apfelmännchen seine Kugeln von sich, die dann zusammenschrumpfen und sich teilend auflösen (s. Bild 6). Am Hauptkörper wächst eine innere Divergenzinsel, wie ein Auge, die wieder nach außen wandert. Unwillkürlich entstehen Assoziationen zum Tierreich. Die fortwährende Teilung der Kugeln erinnert an Zellteilung. Ähnlich behandelt, entwickelt die Gleichung

$$Z = \frac{z + C}{1 + C \times z^k},\ k = 3 \tag{6}$$

aus einem Stern einen kopflosen Schmetterling (Bild 7: a = 0; 0,25; 0,5; 1; 2; 3) oder bildet insektenähnliche Inseln (/5/; Bild 8., a = 0,25).[2]

Zunächst möchte ich feststellen, daß den Computerfraktalen und den natürlich entwickelten Fraktalen derselbe Entstehungsmechanismus zugrunde liegt: Interferenz. So läßt sich vermuten, daß die morphologischen Eigenschaften des biogenetischen Materials auch von schwingenden Ketten bestimmt werden (Makromoleküle). Diese hatten im Urzustand einen symmetrischen Aufbau, waren kleine thermische Mikroschwinger. Durch Mutationen konnte eine Resonanz mit der Umwelt verlängert werden, die auch einen Umbau in Nachbarmolekülen provozierte (frequenzabhängige Abtrennung schwacher Bindungen). Schon hier ist „Vermehrung" (auch Kreuzung) durch Schwingungseffekte anzutreffen. Als Nächstes „überlebten" Molekülgruppen, in deren Interferenz-Knotenflächen sich anorganische Abschirmhüllen anlagern konnten. Später wurden daraus die ersten Zellen und unsere ganze organische Welt.

Ich möchte betonen, daß hier nicht primär die chemischen Schwingungen, wie etwa die DNS-Reproduktion oder Transportgrößen im Sinne der Synergetik, gemeint sind, sondern sehr hohe Eigenfrequenzen, die sich aus der räumlichen Molekülstruktur ergeben. Die Erbinformation ist zwar in chemischer Form kodiert, aber erst die Schwingungsprofile bilden den zweiten, den dynamischen Teil der Information, der sich über die stoffliche Synthese entfaltet und auch diese steuernd (physikalisch) beeinflußt. Denkbar wäre zum Beispiel folgender Vorgang: Ist ein Enzym ausreichend in der Zelle vorhanden, wird über eine intensive Schwingungskopplung auch sein erzeugendes Gen zu Eigenschwingungen angeregt, wodurch die weitere Enzymsynthese gestoppt wird.

Auch ein konstanter Fehler in der Iterationsgleichung muß einer Mutation entsprechen. Bild 4 zeigt eine Mutation von $Z = z^3 + C$, wobei beim Ausmultiplizieren ein Fehler gemacht wurde. Ein innerer Faktor y war aus der Kette

---

[1] Es liegt nahe, verallgemeinerte komplexe Zahlen auch als neue Bewegungs- und Wellenarten zu interpretieren (s. /1/, /2/).

[2] Gleichung (6) ist ein komplexer Faktorregler aus /4/. Diese Regler sind auch als verallgemeinerte Vierpolgleichungen zu verstehen: Eine Kette identischer längssymmetrischer Vierpole transformiert von Glied zu Glied die Ausgangs- und Eingangswiderstände wie z zu Z mit $Z = \frac{R}{W}$, $C = \tanh(g)$, R = komplexer Widerstand, W = Wellenwiderstand, g = Übertragungsmaß
$$Z = \frac{z + C}{1 + z \times C},$$
das heißt für die Potenz $k = 1$, während in (6) $k = 3$ vorliegt.

```
REM  Programm Schmetterling (groesse=10)
DEFINT i,j: DEFDBL a-h,k-z: DIM  r(31),g(31),b(31)
INPUT " Bildgroesse (10..1) : ",groesse  :REM 10 =300*180, 1 =30*30
INPUT "  ab Zeile  (1..180) : ",iy: iy=iy-1   :REM fuer Fortsetzung
 bry=1: brx=bry: vsx=0: vsy=3.95: norm=20: teiler=1  :REM Bilddaten
 param=.25: ianzmax=200: maxabs=100000!: genau=200   :REM Bilddaten
 ispalten=groesse*30: iye=ispalten: IF iye>180 THEN iye=180
 IF groesse>6 THEN brx=brx*ispalten/iye          :REM Bild rechteckig
 swx=brx/ispalten:swy=bry/iye:panf=-brx/2-swx+vsx:REM panf=Abszisse
 tanf=bry/2+swy+vsy-iy*swy: genau=(swx+swy)/genau:REM tanf=Ordinate
SCREEN 1,320,200,5,1: pastr$=STR$(param#)
WINDOW 2,pastr$,(0,0)-(310,180),16,1
    f5=29/31: br=6: y1=0: y2=br: s=1/16: r(1)=1: g(1)=-s
    r(18)=0: g(18)=1: b(18)=0: PALETTE 0,0,0,0: PALETTE 1,1,1,1
   FOR ic=2 TO 31                             :REM Farbkeil senkrecht
    IF ic<18 THEN r(ic)=r(ic-1)-s: g(ic)=g(ic-1)+s: b(ic)=0
    IF ic>18 THEN g(ic)=g(ic-1)-s: b(ic)=b(ic-1)+s: r(ic)=0
    PALETTE ic,r(ic),g(ic),b(ic)
    COLOR ic : LINE(302,y1)-(311,y2),,Bf: y1=y1+br: y2=y2+br
   NEXT:                             PALETTE 18,.6,1,.6
   r(0)=0:g(0)=0:b(0)=0:g(1)=1:b(1)=1:r(18)=.6:g(18)=1:b(18)=.6
WHILE  iy < iye
          iy=iy+1: tanf=tanf-swy: ix=0: panf=-brx/2-swx+vsx: ky$="a"
 punkt:   ix=ix+1: panf=panf+swx: ianz=0: ianzf=0: iL=0
          x=0: y=0: p=x: t=y                     :REM Anfangswerte
 iterat: ianzf=ianzf+1: ky$=INKEY$: IF ky$="s" THEN GOTO endl
     ianz=ianz+1: IF ianz > ianzmax THEN grl!=15: GOTO abbruch
     xalt=x: yalt=y: palt=p: talt=t
                                          :REM "a"=param, C=a+ib
       a=panf*(1+param*p): b=tanf*(1-param*t)       :REM Zwilling 1
     qx=x*x-y*y: qy=2*x*y: q3x=qx*x-qy*y: q3y=qx*y+qy*x
     u=x+a: v=y+b: d=a*q3x-b*q3y+1: e=a*q3y+b*q3x: fsb=d*d+e*e
     IF fsb < 1E-10  THEN   GOTO codel
     xrett=(u*d+v*e)/fsb: yrett=(v*d-u*e)/fsb
       a=panf*(1+param*x): b=tanf*(1-param*y): x=p: y=t   :REM Zw.2
     qx=x*x-y*y: qy=2*x*y: q3x=qx*x-qy*y: q3y=qx*y+qy*x
     u=x+a: v=y+b: d=a*q3x-b*q3y+1: e=a*q3y+b*q3x: fsb=d*d+e*e
     IF fsb < 1E-10  THEN GOTO codel
     p=(u*d+v*e)/fsb:t=(v*d-u*e)/fsb: x=xrett:y=yrett :REM EndeGlg.

  IF ABS(x)+ABS(y)+ABS(p)+ABS(t)>maxabs THEN GOTO codel
  IF ianzf<50 THEN
       dd=ABS(xalt-x)+ABS(yalt-y)+ABS(palt-p)+ABS(talt-t)
       IF dd>genau THEN GOTO iterat
       GOTO code2                                   :REM evtl. grl!=0
    ELSE
       IF iL>0 AND iL<16  THEN
          iL=iL+1: dd=ABS(xxl-x)+ABS(yyl-y)+ABS(ppl-p)+ABS(ttl-t)
          IF dd>genau THEN GOTO iterat
        ELSE
          xxl=x: yyl=y: ppl=p: ttl=t: iL=1: GOTO iterat
       END IF
       grl!=iL: IF iL=2 THEN ianzf=1: GOTO iterat
       GOTO code2                  :REM evtl.GOTO abbruch od. GOTO codel
  END IF
    codel:   grl!=31: GOTO abbruch          :REM evtl. grl!=ianz/teiler
    code2:   grl!=(ABS(x)+ABS(y)+ABS(p)+ABS(t))*norm
             IF grl!>30 THEN grl!=30
             IF grl!<2  THEN grl!=2
    abbruch:
     ic=INT(grl!): PALETTE ic,r(ic),g(ic),b(ic)
     COLOR ic: LINE(ix,iy)-(ix,iy),,Bf
     IF ix < ispalten THEN GOTO punkt
WEND
endl:   ky$="a": WHILE ky$<>"e" : ky$=INKEY$: WEND
ende:   WINDOW CLOSE 2: SCREEN CLOSE 1
END
```

$$X = xxx - xyy - xyy - xyy + Cx$$
$$Y = -yyy + xxy + xxy + xxy + Cy$$

verlorengegangen. Man sieht noch die Verwandtschaft mit Bild 1, aber die „Fühler“ und „Krallen“ sind schon tierischer geworden (vgl. mit /5/). Daraus ist zu folgern, daß die komplexe Wellendarstellung nur den harmonischen Spezialfall beschreibt (kugelige Knospen), während im allgemeinen die Iterationen ein unharmonisches Wellenprofil erzeugen. Erst die Zyklus-Welle als nun übergeordneter, stabiler Schwebungszustand paßt wieder zu unserem bescheidenen Wissen über Wellen und Bewegungen (s. Bild 12: Überlagerung dreier Wellen pro Zeile). In diesem Sinne muß der Begriff Interferenz weiter gefaßt werden: Überlagerung von nichtlinearen (rückgekoppelten) Prozessen, die im stabilen Falle Wellen bilden.

Falls alle Bio-Hierarchien, zum Beispiel die Zellen, solche „Cladnischen Klangfiguren“ darstellen, sollten die Biophysiker versuchen, die Übereinstimmung von Wellenlängen und Strukturgrößen nachzuweisen, beispielsweise in Kombination mit Ultraschallholografie. Möglicherweise sind jedoch elektromagnetische (kein Ladungsgleichgewicht!) oder noch unbekannte Wellen die Entscheidenden.

Das einem Embryo ähnelnde Bild 11 entstammt der Gleichung

$$Z = \frac{z + \frac{C}{z}}{1 + C \times z^3}$$

Abgebildet wird wie immer die C-Ebene mit $z_0 = C$. Die Ring- und Ringelstruktur liegt im Divergenzbereich und betrifft den Abbruch bei großen Z-Werten (15) statt bei kleinen (1 – 10).

Wer sich über erste Grundbegriffe der Chaostheorie informieren will, dem sei /6/ empfohlen. Die Motive in den Bildern 10 und 12 wurden nach der in /1/ beschriebenen Gleichung berechnet, wobei Bild 9 dem Bild 3 aus /1/ mit Ansatz (2) entspricht und Bild 10 mit Ansatz (3) ein Detail der Größe 2 – 3 darstellt.

Für alle Bilder (außer 11) wurde ein Amiga 500 benutzt (32 Farben). Das Listing zeigt eine Minimalvariante des AC-Basic-Programms, hier für Bild 8 (180 Zeilen). Für eine Bildzeile von 300 Punkten werden zirka 9 Minuten benötigt (compiliert). Das Programm entspricht dem in /1/ erläuterten Pascal-Programm, wobei die externe Speicherung im IFF-Format mit Standardprogrammen wie *grabbit* erfolgen muß.

Literatur

/1/ Buhren, G.: Fraktale aus Polynomen. Mikroprozessortechnik, Berlin 2 (1988) 9, S. 260

/2/ Peschel, M.; Buhren, G.: In die Tiefen der Fraktale. spectrum, Berlin 20 (1989) 7/8, S. 32

/3/ Peschel, M.: Morphologische Erscheinungen am Apfelmännchen. spectrum, Berlin 21 (1990) 4

/4/ Peschel, M.: Chaos im Organismus. spectrum, Berlin 20 (1989) 5, S. 18

/5/ Buhren, G.: Fraktale Geometrie – ein Hintergrund des Biologischen? Bild und Ton, Leipzig 43 (1990) 3, S. 80

/6/ Leven, R. W.; Koch, B.; Pompe, B.: Chaos in dissipativen Systemen. WTB-Reihe Mathematik/Physik, Berlin: Akademie-Verlag 1989

Eltern-Kind-Prozesse unter MS-DOS

Wolfgang Thiel, Wismar

Es ist zweckmäßig, oft benutzte Teile von Programmen resident (ständig aktiv) im Hauptspeicher zu belassen. Im vorliegenden Beispiel werden an Hand von C-Komponenten in Verbindung mit Fortran und Pascal Mechanismen des Aufrufes sowie der Parametervermittlung dargelegt.

Das Problem

Die Nutzung von Basiskomponenten ist eine Möglichkeit zur Erhöhung der Effektivität der Softwareentwicklung. Bei größeren Programmsystemen ist es aber unzweckmäßig, diese Komponenten physisch in jedes Teilprogramm einzubinden. Zeitlich wesentlich effizienter ist es, allgemeingültige Module speicherresident zu entwickeln. Das Betriebssystem MS-DOS bietet durch die Methoden der Speicherorganisation (Segmentierung) und die Möglichkeit der Programmverkettung in Form der Eltern-Kind-Prozeßverwaltung dafür eine günstige Ausgangsbasis.

Man darf aber nicht vergessen, daß MS-DOS kein Multitask-/Timesharing-Betriebssystem ist. Es existiert für MS-DOS nur ein Prozeß, der die alleinige Regie hat. Es müssen somit sowohl für die wechselseitige Kommunikation zwischen Eltern- und Kind-Prozeß als auch für die Prozeßumschaltung eigene Lösungen gefunden werden. Im folgenden wird eine solche Lösung dargestellt:

① Der Elternprozeß wird in Turbo-C (Speichermodell *small*) erzeugt. Theoretisch wäre auch eine andere Sprache wie Pascal oder Assembler denkbar. Der Elternprozeß enthält neben einem Organisationsrahmen (u. a. den Aufruf des Kindprozesses) die durch den Kindprozeß zu nutzenden Module.

② Die Kindprozesse können in einer beliebigen Sprache entwickelt werden. Im Beispiel (Bilder 1 bis 5) werden Lösungen für C, Turbo-Pascal (Version 5.0) und Fortran 77 vorgestellt.

③ Die Prozeßumschaltung erfolgt durch einen Softwareinterrupt aus dem aktuellen Kindprozeß heraus.

④ Die Übergabe von Parametern erfolgt zweckmäßigerweise über Register beim Interrupt. Im Beispiel wurde ein Zeiger auf eine Parameterstruktur gewählt. Denkbar sind

```
#include <process.h>
#include <dos.h>
#include <stdio.h>
#include <bios.h>
/*-------------------------------------------------------------*/
/* Variablen, die fuer die Steuerung benoetigt werden          */
/*-------------------------------------------------------------*/
int  far *spmz;               /* Zeiger -> Merkzelle SP       */
int  far *ssmz;               /* Zeiger -> Merkzelle SS       */
extern int far _spawn();      /* Funktion SPAWN               */
/*-------------------------------------------------------------*/
/* Beispiel einer Interruptroutine, die physisch im Elternpro-*/
/* zess liegt und aus dem Kindprozess durch INT gerufen wird. */
/*-------------------------------------------------------------*/
  static int mss,msp,mpsp;    /* Merkzellen fuer SS:SP,PSP    */
  static unsigned far *preg;  /* Zeiger fuer Registerstack    */
  union REGS dos;             /* Register fuer DOS-Ruf        */
  char puffer[256];           /* Zeichenkettenpuffer          */
/*-------------------------------------------------------------*/
/* Beispiel einer Uebergabestruktur mit Varianten              */
/*-------------------------------------------------------------*/
  struct kern {               /* Parameteruebergabe           */
                              /* USER <> System               */
      int  call_kode;         /* Rufkode                      */
      union  variante {
        char far *text;       /* -> Ruf 0 Zeichenkette        */
        int  zahl;            /* -> Ruf 1 Zahl als Wert       */
        struct{               /*,-> Ruf 2 Fortran Zeichenkette*/
          char far *text;     /* Inhaltzeiger                 */
          unsigned int l;     /* Laenge                       */
        } far *fzk;
        int far *zp;          /* -> Ruf 3 Zeiger auf Zahl     */
      } r;
  } far *user;                /* wird auf Kind gestellt       */
/*-------------------------------------------------------------*/
/* Kopieren/Konvertieren von Zeichenketten mit far-Zeigern     */
/*            far_copy - allgemein ,  for_zk - Fortran         */
/*-------------------------------------------------------------*/
  void far_copy(char far *ziel,char far *quelle)
  {
   while (*ziel++ = *quelle++);
  }

  void for_zk(char far *ziel,char far *quelle,unsigned int l)
  {
   while (l--) (*ziel++ = *quelle++);
   *ziel=0;
  }
void interrupt test(unsigned rbp,unsigned rdi,unsigned rsi,
                    unsigned rds,unsigned res,unsigned rdx,
                    unsigned rcx,unsigned rbx,unsigned rax)
{
  preg=&rbp;                  /* Registerzeiger stellen       */
  mss=_SS;      msp=_SP;      /* SS:SP Kind retten            */
  _DX=*ssmz;_CX=*spmz;        /* SS:SP auf Eltern stellen     */
  _SS=_DX;  _SP=_CX;  _BP=_CX;  /* BP standardgemaessig = SP  */
                              /* Parametervektor stellen      */
  user =MK_FP(*(preg+5),*(preg+6));
                              /* Umschaltung PSP              */
  dos.h.ah=81;                /* PSP der Kind retten          */
  intdos(&dos,&dos); mpsp=dos.x.bx;
  dos.h.ah=80;                /* PSP der Eltern einstellen    */
  dos.x.bx=_psp; intdos(&dos,&dos);
/*---------------  Anwerderprogramme  -------------------------*/
/*          SS:SP , BP , PSP stehen auf Elternprozess          */
/*           Verzweiger zu den Anwenderprogrammen              */
/*         oder wie im Beispiel ein kleines Programm           */
/*-------------------------------------------------------------*/
   switch ( user->call_kode ) {
   case  0 :
         far_copy(puffer,user->r.text);
         printf("%20s\n",puffer);
         break;
   case  1 :
         printf("%8d\n",user->r.zahl);
         break;
   case  2 :
         for_zk(puffer,user->r.fzk->text,user->r.fzk->l);
         printf("%20s\n",puffer);
         break;
   case  3 :
         printf("%8d\n",*user->r.zp);
         break;
   }
/*-------------------------------------------------------------*/
/*      Endebehandlung -> Herstellen der Kind-Umgebung         */
/*-------------------------------------------------------------*/
  dos.h.ah=80;                /* PSP Kind einstellen          */
  dos.x.bx=mpsp; intdos(&dos,&dos);

  _DX=mss;  _CX=msp;          /* SS:SP Kind einstellen        */
  _SS=_DX;  _SP=_CX;
}
/*----------------  Ende Interruptroutine  --------------------*/

/*-------------------------------------------------------------*/
/*                Laden des Interruptvektors                   */
/*-------------------------------------------------------------*/
void install(void interrupt(*func)(),int nr)
{
  setvect(nr,func);
}
/*-------------------------------------------------------------*/
/* Einrichten eines Zeigers auf die Merkzellen des Eltern-     */
/* prozesses, in denen SPAWN SS:SP rettet.                     */
/*-------------------------------------------------------------*/
void set_stack_zellen(f)
  int far *f;
{
  spmz=(f-2);                 /* Adresse SPAWN - 4            */
  ssmz=(f-1);                 /*       "      "  - 2          */
}
static  char *C_KIND="CKIND.EXE";
static  char *P_KIND="PKIND.EXE";
static  char *F_KIND="FKIND.EXE";
main()
{
  printf("Elternprocess Start\n");
  set_stack_zellen(_spawn);   /* Zeiger auf Merkzelle stellen */
  install(test,240);          /* Einrichten Interrupt         */
  spawnl(P_WAIT,C_KIND,"");   /* Ruf C-Kind-Prozess           */
  spawnl(P_WAIT,P_KIND,"");   /* Ruf P-Kind-Prozess           */
  spawnl(P_WAIT,F_KIND,"");   /* Ruf F-Kind-Prozess           */
  printf("Elternprocess Ende\n");
}
```

Bild 1 Beispiel Elternprozeß

auch Varianten unter Verwendung globaler Datenbereiche oder auch RAM-Disks bzw. Festplatten.

Das vorgeschlagene Konzept ist kaskadierbar nach dem Prinzip Eltern-Kind-Enkel-.... Damit ist eine Aufteilung der Module in mehrere Teilprogramme, die verkettet aufgerufen werden müssen, möglich. Dabei sollten alle Prozesse, die Interruptroutinen enthalten, nach einem einheitlichen Konzept in C erzeugt werden. Jedem derartigen Prozeß ist mindestens ein Interrupt zuzuordnen. Selbstverständlich kann auch der Kommandoprozessor COMMAND.COM Bestandteil einer derartigen Kette sein.

Für die Entwicklung unter MS-DOS sind folgende Probleme bei der Prozeßumschaltung zu lösen:

- Retten der aktuellen Kindumgebung
- Herstellen der Elternumgebung
- Regenerieren der Kindumgebung.

```
#include <dos.H>
#include <process.H>
#include <bios.H>
#include <stdio.H>
/*-------------------------------------------------------------*/
/*      Parameterblock zum Ruf der Elternfunktionen            */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
 struct kern {                /* Parameterblock               */
      int  call_kode;         /* Rufkode                      */
      union  variante {
            char far *text;   /* --> Ruf 0                    */
            int  zahl;        /* --> Ruf 1 Zahl Wert          */
          } r;
  } auser;
 struct kern far *user=&auser;
 struct REGPACK regis,*preg=&regis;
/*-------------------------------------------------------------*/

main()                        /* Beispielprogramm             */
{
printf("Start C-KIND\n");     /* Startmeldung                 */
regis.r_dx=FP_SEG(user);      /* Parameteradresse             */
regis.r_cx=FP_OFF(user);      /* stellen                      */

user->call_kode=0;            /* Zeichenkettenueberg.         */
user->r.text="'C'-Kind";      /* zum Elternprozess            */
intr(240,preg);               /* Ruf Elternroutine            */

user->call_kode=1;            /* Zahlenuebergabe              */
user->r.zahl=1234;
intr(240,preg);

printf("Ende  C-KIND\n");     /* Endemeldung                  */
}
```

Bild 2 Beispiel C-Kind

```
uses dos;
{-------------------------------------------------------------}
{         Parameterblock zum Ruf der Elternfunktionen         }
{-------------------------------------------------------------}
type
  kern = record                        { Variante call_kode    }
    case call_kode:integer of
      0  : (text : ^char);             { Zeiger auf Text       }
      1  : (zahl : integer);           { Zahl als Wert         }
  end;
{-------------------------------------------------------------}
var user  : kern      ;
    reg   : registers ;                { siehe unit dos        }
    st    : string[20];

begin
  writeln('Start PASCAL-Kind');        { Startmeldung          }
  reg.dx:=seg(user);                   { Zeiger auf Parameter- }
  reg.cx:=ofs(user);                   { block                 }

  user.call_kode:=0;                   { Zeichenkettenausgabe  }
  st := 'PASCAL-Kind'+^@;              { Beachte \0 fuer C     }
  user.text:=addr(st[1]);
  intr(240,reg);                       { INT-Ruf               }

  user.call_kode:=1;                   { Zahlenausgabe         }
  user.zahl:=1234;
  intr(240,reg);

  writeln('Ende PASCAL-Kind');         { Endemeldung           }
end.
```

Bild 3 Beispiel Pascal-Kind

```
c ---------------------------------------
c  !         Fortran 77-Kindprozess        !
c ---------------------------------------
      program FKIND
      print 10
10    format(' Start Fortran 77-Kind '/)
      call elter2('Fortran 77 - Kind')
      call elter3(1234)
      print 11
11    format(' Ende Fortran 77-Kind '/)
      end
```

Bild 4 Beispiel Fortran-Kindprozeß

```
;*************************************************
;  ASM-Schittstellenprogramm fuer Fortran
;*************************************************
       name    FORASM
RUF    macro   name,CALL_KODE,Parameter
       public  name
name:  push    ax
       mov     ax,Parameter * 200h + CALL_KODE
       jmp     RUF1
       endm
code   segment
       assume  cs:code,ds:nothing,es:nothing,
               ss:nothing

fsys   proc    far
;Liste alles Eintrittspunkte
       RUF     elter2,2,1          ;Zeichenkette
       RUF     elter3,3,1          ;Zahl

RUF1:  push    bx
       push    cx
       push    dx
       push    di
       push    si
       push    bp
       push    es
       push    ds                  ;Register retten
       xor     ch,ch
       mov     cl,ah               ;cx=anz Parameter
       push    ss                  ;SS:DI -> Parameter
       pop     ds                  ;in DS:SI umladen
       mov     si,di               ;= Parameter * 4
       add     si,4                ;Aufsatz ab Byte 4
       push    cs                  ;Ziel ES:DI
       pop     es
       mov     di,offset user+2 ;Zeigerposition
       xor     ah,ah
       mov     es:word ptr [di-2],ax
                                   ;CALL-Kode
rep    movsw                       ;laden user
       mov     cx,offset user      ;INT vorbereiten
       mov     dx,seg user
       int     240                 ;Eltern rufen
       pop     ds
       pop     es
       pop     bp
       pop     si
       pop     di
       pop     dx
       pop     cx
       pop     bx
       pop     ax                  ;Register laden
       ret.                        ;retf
user   dw      12 dup(0)           ;CALL-Kode + 5 Zeiger

fsys   endp
code   ends
       end
```

Bild 5 Assemblerunterprogramm für Fortran

Turbo-C gestattet durch den Modifizierer INTERRUPT die Deklaration von Interruptroutinen. Eine genaue Untersuchung der verfügbaren Turbo-C-Version im Speichermodell *small* zeigte jedoch, daß diese Lösung nicht allgemeingültig ist. Die Nutzung anderer Speichermodelle ergab ebenfalls keine verwendbare Variante.

– Die Register SS:SP werden nicht umgeschaltet. Dynamische Speicherbereiche werden aber sowohl über SS:BP als auch über DS:xx adressiert. Das ist nur korrekt, wenn SS und DS den gleichen Inhalt haben.

– Der Programmsegmentpräfix (PSP) wird nicht aktualisiert.

– Einige Funktionen, zum Beispiel PRINTF, arbeiten trotz expliziter Nutzung von FAR-Zeigern intern mit NEAR-Zeigern. Es wird praktisch das eigene Datensegment angenommen und der Segmentteil des FAR-Zeigers ignoriert.

Da die integrierte Turbo-C-Umgebung keine Assembleranweisungen unterstützt, sind diese zu vermeiden. Das ist insofern kein Problem, da der Zugriff auf die Prozessorregister unter Beachtung einiger Restriktionen symbolisch in C erfolgen kann. Abhängigkeiten von Betriebssystem und Hardware sind zu konzentrieren, um eine Portierbarkeit mit geringem Aufwand sichern zu können (z. B. 80286 als Prozessor).

Die Beispiellösung

Das Prinzip soll an einem konkreten Beispiel erläutert werden. Es wird daher von nur einem Elternprozeß ausgegangen, der nacheinander drei Kinderprozesse aufruft, die inhaltlich ähnlich sind, aber in C, Turbo-Pascal und Fortran 77 geschrieben wurden.

Parameterübergabe

Im Prinzip ist es möglich, eine begrenzte Zahl von Parametern direkt über die Prozessorregister zu übergeben. Wesentlich allgemeiner ist jedoch die vorgeschlagene Lösung in Form einer speziellen Datenstruktur. Die Verwendung eines CALL-Kodes erlaubt es, unterschiedliche Programme des Elternprozesses über nur einen Interrupt aufzurufen. Für jeden möglichen CALL-Kode muß eine identische Datenstruktur sowohl im Eltern- als auch im Kindprozeß vereinbart werden. Dazu bieten Turbo-C durch Unions in Strukturen und Turbo-Pascal durch Varianten in Datensätzen ausreichende Mittel. Diese beiden Sprachen unterstützen sowohl die Wert- als auch die Adreßübergabe von Parametern im Stack, wobei Adressen generell als FAR vereinbart werden müssen.

Die Nutzung von Fortran 77 ist etwas komplizierter. Hier wird beim Unterprogrammaufruf ein Vektor vom FAR-Zeiger auf die aktuellen Parameter bereitgestellt. Eine Werteübergabe oder eine Zeigerarbeit wird nicht unterstützt.

Da im Sprachumfang die Auslösung eines Interrupts nicht vorgesehen ist, wurde ein Assemblerprogramm unter Verwendung von Macros entworfen, das den Aufbau der erforderlichen Parameterstruktur und den Interrupt gewährleistet. Dieses Programm kann in die Standardbibliothek eingeführt werden, oder es ist beim Linken explizit anzugeben.

Kompatible Datenformate

Tafel 1 zeigt die nutzbaren Datenformate. Problemlos können die Integer-Formate verwendet werden, Real-Formate sind zwar intern identisch (8087-Formate), aber die zugelassenen Wertebereiche der einzelnen Sprachen sind unterschiedlich. Problematisch hingegen sind Zeichenketten. Es kann jeweils ein Zeiger, der auf eine sprachtypische Struktur zeigt, übergeben werden:

Pascal	Byte 0 = logische Länge ab Byte 1 = Inhalt
C	ab Byte 0 = Inhalt mit folgender 0 als Endekennung
Fortran 77	Adresse einer Verwaltungsstruktur Byte 0...3 = FAR-Zeiger auf Inhalt 4...5 = logische Länge

Tafel 1 Nutzbare Datenformate

Turbo-C	Turbo-Pascal	Fortran 77	Wertebereich
char	shortint		– 128 ... 127
unsigned char	byte		0 ... 255
int	integer	INTEGER*2	– 32768 ... 32767
unsigned int	word		0 ... 65535
long int	longint	INTEGER*4	2147483648 ... 2147483647
float	single	REAL*4	$0{,}29E^{-38} \ldots 3{,}4E^{+38}$
double	double	REAL*8	$0{,}56E^{-308} \ldots 0{,}9E^{+308}$

Zweckmäßigerweise sollte im Elternprozeß eine Konvertierung in das C-Format erfolgen, was günstig mit der notwendigen Kopieroperation verbunden werden kann.

Der Elternprozeß

Das Progammbeispiel in Bild 1 beinhaltet die Hilfsprogramme zum Aufrufen des Kindprozesses sowie eine einfache Demonstration einer Interruptroutine.

MAIN-Funktion

Die spawnl ()-Funktion des Turbo-C-Systems rettet vor dem direkten Start des Kindprozesses über die entsprechende DOS-Funktion die Register SS und SP in Merkzellen, die symbolisch durch *spawn* erreichbar sind. Die Funktion *set_stack_zellen* richtet entsprechende Zeiger ein. Der Kindprozeß kann nach Eintragen der Adresse der Behandlungsroutine (*install*, im Beispiel Interrupt 240) in bekannter Weise mit spawnl gestartet werden. Damit geht die Prozeßregie an den Kindprozeß über. Wird dieser ordnungsgemäß verlassen, wird der Elternprozeß mit der folgenden Anweisung fortgesetzt. Das wird durch MS-DOS organisiert.

Interruptroutine

Durch die Deklaration als Interrupt in Turbo-C werden automatisch die Register AX, BX, CX, DX, ES, DS, SI, DI und BP auf den aktuellen Stack gerettet. Da der Kindprozeß ruft, erfolgt dies in dessen Stack. Anschließend werden das DS-Register mit dem Datensegment des Elternprozesses und BP als Zeiger für die in den Stack geretteten Parameter gestellt. Damit sind die Registerzustände des Interruptrufes als Variablen verfügbar. Da Turbo-C im Speichermodell *small* für seine interne Arbeit DS = SS voraussetzt, ist eine Umschaltung des Stacksegmentes, gefolgt von einer Korrektur von SP, notwendig. Damit sind aber die Parameter der Interrupt-Routine, die über SS:BP des Kindprozesses adressiert werden, nach der Umschaltung nicht mehr verfügbar. Deshalb wird zunächst der Pointer PREG auf die Parameter initialisiert.

Die Umschaltung des Stacks sollte in der angegebenen Reihenfolge ausgeführt werden. Die Verwendung der Register DX und CX ist notwendig, um die Ladeoperation SS,SP als zwei aufeinanderfolgende Befehle erzwingen zu können (Sicherung der Interruptfähigkeit). Weiterhin ist das Register BP korrekt zu stellen. Turbo-C verwendet standardmäßig im C-Modus folgendes Prinzip der Parameterübergabe:

rufendes Programm: Alle Parameter werden in SS:SP gekellert, anschließend wird das zu rufende Programm mit call aufgerufen (FAR oder NEAR sind erlaubt).

gerufenes Programm: Ohne Einrichtung lokaler dynamischer Variablen wird folgender Startkode generiert:

```
PUSH  BP
MOV   BP,SP
```

Damit ist das Register BP als Zeiger auf die Parameter im Stack verwendbar. Die Entfernung der Parameter aus dem Stack ist im C-Modus Sache des rufenden Programms. Das gerufene Programm endet mit:

```
POP   BP
RET          ; NEAR oder FAR
```

Stackkorrektur: Die Entfernung der Parameter erfolgt im rufenden Programm durch den Befehl:

```
MOV   SP,BP
```

Diese Steuerung funktioniert nur, wenn BP auf das Ende des Stacks einer Programmstruktur zeigt.
Bei der als Beispiel angeführten Interruptroutine (Bild 1) mit eigener Stackumschaltung ist also zusätzlich BP mit dem Inhalt von SP zu laden. Lokale Variablen sollten in Interruptroutinen nicht vereinbart werden, da diese nach der Stackumschaltung nicht direkt adressierbar sind und die BP-Verwaltung verkomplizieren.
Die Umschaltung des PSP erfolgt unter Nutzung der MS-DOS-Funktionen 80/81 in bekannter Art und Weise.
Im Beispiel wird eine Parameterüberwachung vom Kind- zum Elternprozeß durch eine spezielle Struktur ermöglicht, deren Adresse durch den Kindprozeß in die Register DX und CX geladen wird. Im Elternprozeß wird der Zeiger *user* auf diese Struktur gestellt. Nach diesen Vorbereitungen können die gewünschten Aktionen ausgeführt werden. In der Endbehandlung werden der PSP sowie SS und SP des Kindprozesses eingestellt (beachte die Verwendung von DX, CX).
Die POP-Operation für die Register sowie der IRET werden durch Turbo-C automatisch generiert (Interrupt-Deklaration). Da Turbo-C nicht korrekt mit FAR-Zeigern arbeitet, sollten entweder eigene Funktionen benutzt werden, oder die Daten sollten, wie am Beispiel einer Zeichenkette gezeigt, in Puffer des Elternprozesses kopiert werden (far-copy).
Weiterhin ist zu beachten: In der Interrupt-Routine dürfen aufgrund der Stackumschaltung keine lokalen Variablen deklariert werden. Es ist zweckmäßig, eine globale Deklaration als *static* zu verwenden. Diese einfache Methode ist jedoch nicht rekursiv verwendbar. Ist diese jedoch gefordert, so müssen die Merkzellen für die Register SS, SP und den PSP sowie die Zeiger USER und PREG stackartig organisiert werden.

Die Kindprozesse

Inhaltlich sind die drei Varianten in den Bildern 2 bis 4 ähnlich. Es wird eine Interruptroutine im Elternprozeß durch den aktuellen Kindprozeß zweimal gerufen. Zunächst soll eine Zeichenkette des Kindprozesses, die durch einen Zeiger adressiert ist, mit einer PRINT-Anweisung des Elternprozesses ausgegeben werden. Durch die Zeiger ist das in C und Pascal recht einfach, wobei in diesem Fall in Pascal ein C-gerechtes Format organisiert wird. Für Fortran 77 wird die Konvertierung ins C-Format im Elternprozeß vorgenommen. Danach wird eine Integerzahl vom Kind- zum Elternprozeß übergeben. Das erfolgt in C (Bild 2) und Pascal (Bild 3) als Wert und in Fortran 77 (Bilder 4 und 5) als FAR-Zeiger. Diese einfachen Beispiele sollen als Anregung dienen. Für eigene Lösungen ist generell zu beachten:
– Die Struktur zur Parameterübergabe des Kindprozesses muß zur Vereinbarung im Elternprozeß kompatibel sein.
– Es dürfen nur FAR-Zeiger/-Adressen verwendet werden. NEAR-Zeiger führen zwangsläufig zu Fehlern.
– Es können bei Verwendung unterschiedlicher Sprachen nur kompatible interne Formate verwendet werden. Im Beispiel wird das für die Zeichenkette in Pascal durch explizites Anfügen von \0 erreicht.
– Fortran übergibt FAR-Adressen. Werte können nicht direkt übermittelt werden.

⊠ KONTAKT ☏
Technische Hochschule Wismar, Sektion M/N, Wissenschaftsbereich Informatik, Philipp-Müller-Straße, Wismar, 2400

Grafik mit dem EC 1834

Andreas Bliesener, Rostock

Die Turbo-Pascal-Compiler der Versionen 4.0 und 5.0 stellen leistungsfähige Bibliotheken zum Programmieren grafischer Ausgaben zur Verfügung. Es sollen hier einige Aussagen zur Nutzung dieser Software für den Rechner EC 1834 mit Grafik-Adapter gemacht werden.

Der EC 1834 besitzt ebenso wie der A 7150 einen Grafik-Adapter auf Basis des GDC U 82720. Der EC 1834 emuliert dabei eine CGA-Karte, deren grafische Fähigkeiten eher bescheiden zu nennen sind.
Durch die Bereitstellung des Treibers VE-89.SYS werden zwei neue Grafik-Modi mit erheblich größerer Leistungsfähigkeit verfügbar. Der Modus 8 ermöglicht zwar die Darstellung von 640 × 480 Punkten in 16 Farben, aufgrund des verwendeten Grafikprozessors wird jedoch keine Kompatibilität zum VGA-Standard erreicht.
Der Modus 9 kann 640 × 400 Punkte in zwei Farben darstellen. Dieser Modus ist kompatibel zum Grafik-Adapter von AT & T. Leider ist aufgrund der unterschiedlichen Initialisierung dieses Modus eine Benutzung der für die Turbo-Pascal-Compiler vorhandenen Bibliotheken nicht ohne weiteres möglich. Die Benutzung der Unit Graph und der Graphix-Toolbox ist aber dennoch möglich.

Die Unit Graph und der Video-Modus 9

Die Unit Graph benutzt als Schnittstelle zwischen allgemeinen Unterprogrammen für die Grafik und dem Video-Adapter das sogenannte Borland Graphics Interface (BGI). Dieses Interface besteht aus Treiber-Programmen, die in Form von Dateien mit dem Suffix *BGI* vorliegen. Dieses Konzept ermöglicht das Erstellen von Programmen, die auf den unterschiedlichsten Video-Adaptern (CGA, EGA, VGA, Hercules) ohne große Änderungen lauffähig sind.
Der Video-Modus 9 des EC 1834 aber läßt sich durch keinen der vorhandenen BGI-Treiber direkt ansprechen. Durch einen Patch in dem BGI-File ATT.BGI ist dies jedoch leicht möglich.
Die Auflösung von 640 × 400 Punkten wird, wie schon erwähnt, durch den Modus 9 initialisiert. Im Original-Treiber geschieht dies jedoch durch den Modus $40.
Das folgende Programm tauscht deshalb die Modus-Nummer und ermöglicht dadurch die richtige Initialisierung des Bildschirms (Bild 1).
Vor der Benutzung des Treibers, muß er jedoch explizit in folgender Form angemeldet werden.

```
VAR   GraphDriver,
      GraphMode   : INTEGER;

BEGIN
  GraphDriver:=ATT400
  GraphMode:=ATT400hi;
  .
  .
END.
```

Eine Initialisierung in der Form

GraphDriver:=Detect; {automatische Erkennung}

funktioniert nicht, da die Grafikkarte mit 256 KByte RAM ausgestattet ist und als EGA/VGA erkannt wird.

Die Graphix-Toolbox (Version 4.0)

In der Graphix-Toolbox wird ebenfalls eine Trennung in hardwarenahe und die eigentlichen Zeichnungsroutinen vorgenommen. Die hardwarenahen Routinen werden bei der Installation der Graphix-Toolbox in die Unit GDRIVER eingebunden.

Bild 1 Patch-Programm für den Treiber ATT.BGI

```
PROGRAM Patch;     {EC 1834  - Turbo 5.0 ATT.BGI vom 10.10.88}
                   {           Turbo 4.0 ATT.BGI vom 09.12.87}
USES CRT,DOS;
{$DEFINE TURBO5} {Definiere Constante für Turbo 5.0 ATT.BGI}
                    {Für Turbo 4.0 ATT.BGI $DEFINE entfernen}
{$IFDEF TURBO5}
  CONST Adr = 1563;       {Adresse des zu verändernden Bytes}
{$ELSE}
  CONST Adr = 1365;
{$ENDIF}

VAR BGI   : FILE OF BYTE;
    B     : BYTE;
    Datum : LONGINT;
    Dat   : DATETIME;

BEGIN
  Assign(BGI,'ATT.BGI');
  Reset(BGI);
  GetFTime(BGI,Datum);                 {Hole Filedatum und -zeit}
  UnPackTime(Datum,Dat);
  ClrScr;
  WriteLN('Der Treiber ATT.BGI vom ',Dat.Day:2,'.',
          Dat.Month:2,'.',Dat.Year,
         ' wird an das Video-System des EC 1834');
  WriteLN('angepaßt.');
  B:=$09;          {Der Wert für den gewünschten Video-Modus}
  Seek(BGI,Adr);
  Write(BGI,B);
  SetFTime(BGI,Datum);         {Setze altes Filedatum und -zeit}
  Close(BGI);
  WriteLN;
  WriteLN('Fertig.');
END.
```

```
Zeile 172:
const
  GrafMode = $0009; { BIOS Interrupt 10 AX register }
                    { Im normalen ATT-Driver steht hier $0040 }

Zeile 296:
  DisplayType:=IBMCGA; {Einfugen -
                        Es wird eigentlich eine EGA erkannt.
                        Im weiteren Programmverlauf wurde
                        bei einer EGA die Farbpalette ver-
                        ändert. }
```

Bild 2 Initialisierung des Video-Modus

```
Zeile 548:
var picture : screentype;

procedure SaveScreen{(FileName : WrkString)};
type PicFile = file of ScreenType;
var PictureFile : PicFile;
    IOErr       : boolean;

procedure IOCheck;
begin
  IOErr := IOresult <> 0;
  if IOErr then
    Error(27, 5);
end; { IOCheck }

begin
  if FileName <> '' then
    begin
      IOErr := false;
      move(ptr(grafbase,0)^,picture,ScreenSizeGlb shl 1);
      Assign(PictureFile, FileName);
      {$I-} Rewrite(PictureFile); {$I+}
      IOCheck;
      if not IOErr then
      begin
        {$I-} Write(PictureFile, Picture); {$I+}
        IOCheck;
      end;
      if not IOErr then
      begin
        {$I-} Close(PictureFile); {$I+}
        IOCheck;
      end;
    end
  else
    Error(27, 5);
end; { SaveScreen }
```

Bild 3 Änderung der Prozedur SaveScreen

```
procedure LoadScreen{(FileName : WrkString)};
type
  PicFile = file of ScreenType;
var
  PictureFile : PicFile;
begin
  if FileName <> '' then
    begin
      Assign(PictureFile, FileName);
      {$I-} Reset(PictureFile); {$I+}
      if IOresult <> 0 then
        Error(11, 5)
      else
        begin
          Read(PictureFile, Picture);
          Close(PictureFile);
          move(picture,ptr(grafbase,0)^,ScreenSizeGlb shl 1);
        end;
    end
  else
    Error(11, 5);
end; { LoadScreen }
```

Bild 4 Änderung der Prozedur LoadScreen

Für den AT & T-Adapter ist auch hier die Möglichkeit gegeben, ihn an den EC 1834 anzupassen.
Alle folgenden Änderungen betreffen das File GRAFATT.DRV (Bild 2).
Die Prozeduren in Bild 3 ersetzen die originalen Prozeduren Save-Screen und LoadScreen, da diese am EC 1834 zum Rechnerabsturz führten.
Nach dem Eintragen der Änderungen wird eine kompilierfähige Version der Graphix-Toolbox mit der Batch-Datei TGINST.BAT erstellt.
Der Aufruf lautet:
[Laufwerk:] [Pfad] \ > TGINST ATT.
Danach kann das Programm TGDEMO.PAS kompiliert werden, wobei die Units GDRIVER, GKERNEL, GWINDOWS und GSHELL erzeugt werden. Damit ist eine Nutzung der Graphix-Toolbox im vollen Leistungsumfang möglich.

Literatur

/1/ Turbo-Pascal 4.0 Referenzhandbuch Heimsoeth & Borland 1987

✉ KONTAKT ☎

Institut für Schiffbautechnik, Am Strom 109, Rostock, 2530;
Tel. 56 22 5

L3, PC-MOS, VM/386

3 Systeme, 3 Methoden

David Possin

Wer noch mit MS-DOS arbeitet, nutzt die Leistung eines 32-Bit-Personal-Computers bei weitem nicht aus. Steigt man auf Unix um, läßt sich meist die vorhandene DOS-Software nicht mehr nutzen. Den Ausweg aus dem Dilemma versprechen DOS-kompatible Multiuser/Multitasking-Systeme wie L3, PC-MOS oder VM/386. (pl)

Derzeit ist MS-DOS in der Praxis bei PC-Betriebssystemen Nummer eins der Weltrangliste. An dieser Position dürfte sich in den nächsten Jahren trotz OS/2 oder Unix auch nichts Entscheidendes ändern.
Microsoft und IBM versuchen zwar mit OS/2 einen würdigen Thronfolger für DOS zu küren, aber die Komplexität und die damit verbundenen Probleme wie Programmfehler, enormer Speicherbedarf und fehlende Zahl von Standardprogrammen verhindern bis jetzt den erhofften Durchbruch.
Sicherlich wird OS/2 als Multitasking-Betriebssystem auch erst seine Bedeutung als Multiuser-System erhalten, wenn es erst einmal die Speicherverwaltungs-Techniken der 80386- und 80486-Prozessoren unterstützt und entsprechende Zusätze erhältlich sind, damit mehrere DOS-Tasks auch von Terminals angesprochen werden können. Die Entwicklung des LAN-Managers für OS/2 zeigt schon in die genannte Richtung. Nur: Bis dies soweit ist, nutzen die Hersteller anderer Betriebssysteme die Lücke aus, die entstanden ist. Und diese Lücke scheint ziemlich groß zu sein, wenn man die Anzahl der Betriebssysteme und der Betriebssystem-Erweiterungen betrachtet, die die nächsten Schritte nach DOS darstellten.
Unix dagegen hat im Prinzip durch seine technischen Vorteile und die große Akzeptanz in der mittleren Datentechnik die besseren Karten. Nur entspricht sein vorgesehenes Einsatzgebiet als Mehrplatz-Betriebssystem nicht dem Bedarf nach einem neuen PC-Standardsystem: Der DOS-Standard darf nicht ignoriert werden. Und dieses Standardsystem muß durchgängig als Einplatz-Grundsystem über Multitasking bis hin zum Mehrplatzsystem homogen ausbaubar sein, auch innerhalb von Netzwerken.
Wie weit dies heute schon möglich ist, zeigen Betriebssysteme wie »L3«, »PC-MOS« und »VM/386«.
VM/386 vertritt die Philosophie, DOS bleibe das Maß der Kompatibilität und sei nicht ohne weiteres zu ersetzen. PC-MOS nimmt für sich eine sehr hohe Kompatibilität in Anspruch, ohne aber zu abhängig von DOS zu werden. L3 möchte den 80386-Prozessor möglichst voll ausnutzen und bietet als Bonbon eine gewisse DOS-Kompatibilität. L3 ist eine Entwicklung der deutschen Firma Ergos, die aus der Gesellschaft für Mathematik und Datenverarbeitung (GMD) entstand. Laut Systemdefinition war bei L3 die Kompatibilität nicht so entscheidend, statt dessen steht maximale Leistung obenan.
Die Entwickler von L3 betonen, daß DOS-Kompatibilität sich vorrangig auf die Dateiformate bezieht, DOS-Dateien können von Disketten gelesen und beschrieben werden. Viele DOS-Programme sind unter L3 lauffähig, bestimmte grundsätzliche Einschrän-

Auf einen Blick: Ergos L3

Programmart: PC-Betriebssystem
Bedienung: Tastatur
Leistungsmerkmale: Multitasking, Multiuserfähigkeit, teilweise DOS-Kompatibilität
Preise: 1014,60 DM für 1 Benutzer,
2736 DM für max. 4 Benutzer,
4503 DM für theoretisch unbegrenzte Benutzerzahl (Preis nach Herstellerangabe inkl. MwSt.)
Testeindruck:
+ schnelles Betriebssystem
+ theoretisch unbegrenzte Benutzerzahl
+ besonders für Spezial-Anwendungen geeignet
– eingeschränkte DOS-Kompatibilität

Auf einen Blick: VM/386

Programmart: PC-Betriebssystem
Bedienung: Tastatur
Leistungsmerkmale: Multitasking, Multiuserfähigkeit, DOS-Kompatibilität
Preise: 1700 DM für theoretisch unbegrenzte Benutzerzahl
(Preise nach Herstellerangabe inkl. MwSt)
Testeindruck:
+ nahezu vollständige DOS-Kompatibilität
+ theoretisch unbegrenzte Benutzerzahl
+ sehr flexibel einstellbar
– hoher Speicherbedarf

Auf einen Blick: PC-MOS/386

Programmart: PC-Betriebssystem
Bedienung: Tastatur
Leistungsmerkmale: Multitasking, Multiuserfähigkeit, DOS-Kompatibilität
Preise: 786,60 DM für 1 Benutzer,
2074,80 DM für max. 5 Benutzer
3249 DM für max. 25 Benutzer
(Preise nach Herstellerangabe inkl. MwSt)
Testeindruck:
+ sehr hohe DOS-Kompatibilität
+ deutsche und englische Version
+ insgesamt einfache Handhabung
– Installation durch Spezialisten ratsam

kungen sind aber zu beachten. Die Festplatte wird von L3 mit einem eigenen Format eingerichtet, eine DOS-Plattenpartition kann erhalten bleiben, um gegebenenfalls auch DOS booten zu können. Die Installation über Menüs wirft keine großen Probleme auf. Hardware-Voraussetzungen sind ein 80386-Prozessor, mindestens 2 MByte, besser 4 MByte RAM und eine schnelle Festplatte. L3 arbeitet mit dem Speicher im Paging-Verfahren, das heißt, wenn der Hauptspeicher nicht ausreicht, werden Teile des RAMs auf die Platte ausgelagert. Nach dem Booten erscheint das L3-Systemmenü, das die Tasksteuerung übernimmt. Hier können Tasks angelegt, verwaltet und wieder entfernt werden. Diese können Verwaltungstasks, Tasks für mehrere lokale Anwendungen oder Tasks zur Unterstützung von Terminals, also für den Multiuser-Betrieb, sein. Am besten werden Programme laufen, die in der im Lieferumfang enthaltenen Programmiersprache Elan geschrieben wurden. Wenn ein DOS-Programm gestartet werden soll, muß erst die Systemumgebung hierzu geladen werden. In einem Handbuch-Kapitel steht hierzu ausführlich, welche DOS-Befehle und -Aufrufe bereits implementiert wurden, welche in Arbeit sind und welche nicht übernommen werden. Die Tasksteuerung besitzt sehr umfangreiche Schnittstellen, um Daten von einer Task zur anderen zu schicken. Eine spezielle Task »POST« dient, wie der Name schon sagt, als elektronischer Briefkasten, damit sich die verschiedenen Anwender gegenseitig Briefe schicken können. Eine weitere Task dient als Spooler, um die gemeinsame Nutzung des zentralen Systemdruckers zu verwalten. Bis zu 15 vorgegebene serielle Schnittstellen können verwaltet werden, somit stehen maximal 16 Arbeitsplätze zur Verfügung. Weil die Adressen und Interrupts der Schnittstellen voreingestellt sind, muß bei der Auswahl der Schnittstellenkarten auf diese Eigenschaften geachtet werden. Als Terminals sind die sogenannten PCTERM-fähigen geeignet, die neben den 25 Bildschirmzeilen auch den erweiterten IBM-Zeichensatz vertragen.

Ein integriertes Archivierungssystem verwaltet sowohl die Disketten-Laufwerke als auch Bandlaufwerke, um Konflikte bei gleichzeitigen Zugriffen auf diese Speichermedien zu unterbinden. Zahlreiche Hilfsprogramme geben dem Systemverwalter eine Übersicht der Auslastung des Systems, die Aktivitäten der einzelnen Tasks und deren Berechtigungen. Andere Befehle entsprechen in ihrer Funktion mehr oder weniger DOS-Befehlen, aber deren Syntax erinnert mehr an eine Programmiersprache. Die L3-eigene Sprache Elan wird kurz im Handbuch beschrieben. Elan wird an vielen Universitäten als Mustersprache für die strukturierte Programmierung benutzt.

Ingesamt macht L3 einen stabilen Eindruck und ist für spezielle Einsatzgebiete gut zu verwenden. Seine eingeschränkte DOS-Kompatibilität grenzt jedoch den Anwenderkreis ein, aber neuere Versionen sollen auch in diesem Bereich stark verbessert werden. Es ist etwas für Spezialisten, die spezielle Anwendungen zu lösen haben und ein schnelles Betriebssystem brauchen.

PC-MOS, das modulare Betriebssystem

PC-MOS setzt ganz andere Prioritäten als L3. Einmal installiert, soll beim Anwender das Gefühl geweckt werden, er arbeite ganz normal unter DOS. Entsprechend identisch zur Vorlage sind auch die Befehle, die genau so heißen und sich fast immer genauso verhalten. Wer will, kann tiefer einsteigen und die zusätzlichen Möglichkeiten, die ein Multitasking-/Multiuser-Betriebssystem automatisch mit sich bringt, ausnutzen. Die Kompatibilität zu DOS ist sehr hoch, selbst Programme, die DOS-Teile aus Performance-Gründen umgehen, werden erfolgreich abgefangen und »umgebogen«. Die Installation ist nicht einfach und sollte grundsätzlich von einem systemerfahrenen Spezialisten vorgenommen werden. Es sind sehr viele hardwareabhängige Eigenschaften einstellbar, die wichtig für eine gute Leistung und für den sicheren Betrieb sind. Wenn die Installation einschließlich der Batchdateien für die einzelnen Tasks erfolgt ist, wird im normalen Betrieb selten ein Systemverwalter benötigt, höchstens für die Überwachung der Sicherungen und der Systemdrucker. Läuft das System erst einmal stabil mit allen Anwendungen, fühlen sich alle Benutzer wie auf einem PC zu Hause.

Natürlich sind hier die gleichen Restriktionen wie bei allen Mehrplatzsystemen zu beachten. Spielprogramme und Programme, die in ein System direkt eingreifen und dort Manipulationen durchführen, sind absolut tödlich. Hält man sich aber an die üblichen DOS-Richtlinien, und hat entsprechende Sicherheitsvorkehrungen getroffen, um unerlaubte und unerwünschte Eingriffe zu unterbinden, dann hat man ein echtes DOS-kompatibles Mehrplatzsystem.

PC-MOS kann in drei Varianten bezogen werden; als Einplatzsystem mit Multitasking, als Fünfplatzsystem und als 25-Platzsystem. Die Arbeitsplätze beziehen sich nicht auf die Anzahl der Tasks, insgesamt sind 100 Tasks möglich, also mehrere Tasks pro Arbeitsplatz sind realisierbar. Alle Versionen können in ein Novell-Netzwerk eingebunden werden, Novell liefert hierzu die entsprechenden Treiber. Dadurch können sogenannte hybride Netze aufgebaut werden. Dazu wird innerhalb einer Abteilung ein Multiuser-System installiert. Die einzelnen Abteilungen werden untereinander über ein Netzwerk gekoppelt. Gegebenenfalls können einzelne Arbeitsplätze, sofern diese technisch entsprechend ausgestattet sind, parallel hierzu an einen Mainframe-Rechner gekoppelt werden, oder dieser Anschluß erfolgt zentral über einen im Netzwerk befindlichen Rechner, der speziell für diese Aufgabe ausgestattet ist.

VM/386, DOS mit neuem Gewand

Wie bereits beschrieben, braucht bei VM/386 das gewohnte Betriebssystem DOS nicht im Schrank zu verschwinden, sondern wird in jeder Task als Basis hergenommen, um Programme ablaufen zu lassen. VM/386 setzt bezüglich der Festplatte keine Besonderheiten voraus, sondern wird nach dem Booten von DOS erst aktiviert. Zuerst übernimmt er dann die Verwaltung des Prozessors und geht in den echten 386er-Betrieb über. Wenn keine Automatismen angelegt wurden, erscheint ein Menü für den Systemverwalter, um alle möglichen Operationen durchzuführen. Dabei sind unter anderem natürlich Task für Anwendungen anzulegen. Diese Tasks sind sehr fein für den aktuellen Bedarf einstellbar und interaktiv auch dynamisch veränderbar. Schaltet man in eine DOS-Tasks um, so erscheint diese wie eine echte DOS-Umgebung, die sie durch das Laden von DOS auch wirklich ist. In jeder Task wird DOS individuell gebootet mit allen bekannten Merkmalen, was jederzeit von jedem Anwender wiederholt werden kann. Die Installation von VM/386 ist einfacher als bei PC-MOS. Die individuelle Anpassung der Tasks ist jedoch aufwendiger. Steht das System, dann ist das VM/386-System ebenfalls sehr stabil und noch DOS-kompatibler als PC-MOS. Dafür ist es insgesamt etwas langsamer und verbraucht wesentlich mehr Speicher. Beide Systeme sind als Einplatz- oder Mehrplatz-System lieferbar, die Einbindung in ein Netzwerk ist mit Zusatzmodulen ebenfalls bei beiden möglich. Bei steigender Anzahl von Arbeitsplätzen fällt die Leistungskurve von PC-MOS bei bis zu drei Plätzen stärker ab, danach sinkt die Leistung unerwartet schwächer.Bei VM/386 reduziert sich die Leistung gleichmäßiger pro Platz.

Bei PC-MOS und VM/386 ist es sehr schwer, dem einen oder anderen den Vorzug zu geben. PC-MOS wird auch in einer deutschen Version geliefert und bietet in seiner insgesamt einfacheren Handhabung sicherlich die bessere Lösung bei Installationen, die kommerzielle Anwendungen unterstützen sollen. Allerdings sollte vorher die Kompatibilität überprüft werden, da hier möglicherweise Probleme entstehen können. Wenn ein System jedoch häufig umkonfiguriert werden muß, die Anwendungsbereiche nicht eindeutig überprüfbar oder definierbar sind, dann ist VM/386 die bessere Wahl. Die Tasks und deren individuelle Bedürfnisse sind sehr fein zu optimieren. Wenn CAD-Anwendungen oder ähnlich aufwendige Programme häufiger benutzt werden sollen, dann ist die verringerte Leistung ein Preis, der zugunsten der Flexibilität bezahlt werden muß.

In der grundsätzlichen Frage, ob kompatibel oder nicht, ist Ergos L3 eingeschränkt und PC-MOS sehr weitgehend zuverlässig. VM/386 erfüllt den eigenen Anspruch aus DOS-Kompatibilität nahezu vollständig.

Nachdruck aus Computer Persönlich, Heft 5/1990

Turbo-Pascal-Praxis

Teil 6

Manfred Zander, Dresden

Liebe Leserin, lieber Leser,
nachdem Sie unserem Kurs Turbo-Pascal-Praxis nun schon über fünf Folgen und fast ein Jahr die Treue gehalten haben, und hoffentlich auch zahlreiche Hinweise, Tips und Kniffe für Ihre tägliche Arbeit entnehmen konnten, lesen Sie heute unseren sechsten und letzten Teil.
Wir beenden zunächst die Abhandlungen zu den Baumstrukturen und bieten Ihnen dann neben einem Beispiel für die Programmierung von Eingabemasken das vollständige Listing eines kleinen Texteditors an, den Sie sicher sehr gut in Ihren eigenen Programmen nutzen können.

Das Löschen eines einzelnen Elementes einer Knotenliste ist nicht vorgesehen, da eine Liste später immer als eine Einheit, als komplettes Ergebnis einer Opteration betrachtet werden soll. Demgegenüber ist es natürlich notwendig, das Löschen von kompletten Listen vorzusehen. Diese Aufgabe wird von der Prozedur *LöschenKnotenListe* wahrgenommen (vgl. Bild 51, MP 3/90, S. 82). Dazu wird ihr der Zeiger auf die zu löschende Liste übergeben. Nachdem die Prozedur die Elemente der übergebenen Knotenliste vollständig freigegeben hat, wird über diesen Zeiger der Wert NIL zurückgegeben, die Liste existiert nicht mehr.

Zur Auswertung aufgebauter Knotenlisten sind zwei Funktionen und eine Prozedur erforderlich. Die Prozedur *DruckeKnotenListe* zeigt den Inhalt einer Liste auf dem Bildschirm an. Sie setzt ihrerseits eine Prozedur mit Namen *DruckeKnoten* voraus, da für verschiedene Knoteneintrag-Typen die Bildschirmanzeige unterschiedlich sein kann. Die Prozedur *DruckeKnotenListe* ist daher darauf beschränkt, alle in der Liste gesammelten Knoten für die Anzeige bereitzustellen.

Mit der Funktion *Anzahl* ist es möglich, die Elementezahl einer übergebenen Knotenliste zu ermitteln. Sie durchläuft die Liste von Anfang bis zum Ende und zählt dabei die einzelnen Knotenverweise zusammen. Die Liste selbst wird dabei nicht verändert; zurückgegeben wird die Anzahl als Integerwert.

Zum Test, ob ein Knoten in einer Knotenliste enthalten ist, wurde die Boolesche Funktion *Member* erstellt, die die Funktion *gleich* nutzt (s. Prozedur *FindeKnoten*).

Nachdem die Verwaltung der Ergebnis-Knotenlisten den vorgestellten Prozeduren anvertraut wurde, kann zur Baumanalyse übergegangen werden. Alle im Bild 52 vorgestellten Analyseprozeduren übergeben als Ergebnis eine Knotenliste, in der die jeweils ermittelten Knoten gesammelt sind. Zur Erläuterung der einzelnen Analysen soll wieder die Organisationsstruktur eines Betriebes dienen, wobei in jedem Knoten jeweils eine Person gespeichert sein soll.

Die erste abgedruckte Analyseprozedur mit dem Namen *ZweiginListe* sammelt alle einem Knoten untergeordneten Knoten in einer Liste, einschließlich des Knotens selbst. Auf einen Abteilungsleiter angewandt, enthält die Knotenliste beispielsweise alle Gruppenleiter und alle Mitarbeiter der Abteilung. Für den Betriebsdirektor enthält die entstehende Liste alle Betriebsangehörigen. Die wiederum rekursive Prozedur geht dabei folgendermaßen vor:

Zuerst wird der Knoten, auf den der Parameter B zeigt, in die zu erstellende Liste aufgenommen. Dies wäre in unserem Fall der Abteilungsleiter. Anschließend ruft sich die Prozedur für alle in der *Nach*liste gespeicherten Knoten selbst auf, also für alle Gruppenleiter. Bei der Bearbeitung der Gruppenleiter trägt die Prozedur diese wieder in die Liste ein und ruft sich dann weiter für die Mitarbeiter auf. Durch dieses rekursive Prinzip ist gewährleistet, daß alle an einen Knoten – hier den Abteilungsleiter – anschließenden Zweige erreicht werden. In der entstehenden Liste sind dann alle Personen der Abteilung vom Abteilungsleiter selbst bis zum einzelnen Mitarbeiter enthalten. Die Reihenfolge der Listeneintragungen ist dabei willkürlich und für die weitere Auswertung kaum informativ. Die Prozedur *ZweiginListe* übergibt über den Parameter *Liste* den Zeiger auf die erstellte Knotenliste, die nun zur weiteren Verarbeitung, beispielsweise zur Ausgabe, bereitsteht. Die nächste Analyse-Prozedur mit dem Namen *UnterinListe* sammelt dagegen nur die einem speziellen Knoten B (zum Beispiel dem Abteilungsleiter) direkt untergeordneten Knoten (die Gruppenleiter) in einer Liste. Die dabei entstehende Knotenliste entspricht völlig der *Nach*liste *nL* des Abteilungsleiters. Im Unterschied zu der im Baum verankerten *Nach*liste kann die entstandene Knotenliste aber beliebig geändert werden, ohne den Baum zu *beschädigen*. Die Prozedur *UnterinListe* übergibt über den Variablenparameter *Liste* wieder einen Zeiger auf die erarbeitete Knotenliste.

Die Prozedur *AlleEndinListe* ist rekursiv aufgebaut und ähnelt der Prozedur *ZweiginListe*. Die entstehende Liste enthält aber nur *die* Mitarbeiter, die selbst keine Unterstellten mehr haben. Wird der Prozedur über den Parameter B der Zeiger auf einen Abteilungsleiter übergeben, so werden in der Knotenliste alle Mitarbeiter der Abteilung ohne Leitungsfunktion gesammelt. Zeigt B auf den Betriebsdirektor, so werden alle Mitarbeiter des Betriebes, die keine Leitungsfunktion ausüben, in die Liste aufgenommen. Durch das rekursive Vorgehen der Prozedur ist wiederum gesichert, daß alle Verzweigungen des Baumes berücksichtigt werden.

Die letzte vorgestellte Analyse erhielt den Namen *RückinListe*. Sie ermittelt den Weg von einem übergebenen Knoten P zurück zum Kopfknoten des Baumes. Dabei speichert sie alle Knoten, die auf diesem Weg liegen, in einer Knotenliste ab. Zeigte der Parameter P zum Beispiel auf einen Gruppenleiter, so enthält die entstehende Knotenliste lediglich seine Leiter bis hin zum Betriebsdirektor. Auch diese Prozedur übergibt dem rufenden Programm einen Zeiger auf die entstandene Ergebnisliste.

Weitere Analysen könnten beispielsweise folgende Aufträge erledigen: Ermittlung aller Leiter eines Betriebes; Ermittlung aller Leiter der gleichen Leitungsebene; beides für den gesamten Betrieb oder auf Bereiche eingegrenzt; Ermittlung des korrekten Dienstweges zwischen zwei vorgegebenen Personen in der Struktur usw. Solche und weitere Aufträge konkreter Anwendungen sind im Rahmen der vorgestellten Methodik leicht zu erarbeiten.

Zur Demonstration der Arbeit mit einem beliebig verzweigten Baum und den vorgestellten Analysenprozeduren dient wieder ein kleines Programm (Bild 53). In diesem Beispiel werden die in den Bildern 50 bis 52 abgedruckten Teilkomplexe über Include-Anweisungen eingefügt. Die Reihenfolge der Einfügungen darf dabei nicht verändert werden. Auch die Stellung der Funktion *gleich* und der Prozedur *DruckeKnoten* im Verhältnis zu den Einfügungen ist nur so möglich. Der Anweisungsteil dieser beiden Unterprogramme dagegen wird von Lösung zu Lösung unterschiedlich sein, und ist speziell

```
(******************************)
(* Include Baumanal.INC       *)
(******************************)

{$A-}   (* Erzeugen rekursiven Codes !!!!! *)
procedure ZweiginListe(B:Knoten;var Liste:Verweis);
  var V:Verweis;                      (* Sammeln aller untergeordnetten *)
  begin DazuKnotenListe(Liste,B);     (* Knoten in einer Liste          *)
    V:=B^.nL;
    while V<>NIL do
      begin ZweiginListe(V^.nach,Liste);V:=V^.next
  end end;
procedure UnterinListe(B:Knoten;var Liste:Verweis);
  var V:Verweis;                     (* Sammeln aller direkt unterge-   *)
  begin V:=B^.nL;                    (* ordnetten Knoten in einer Liste *)
    while V<>NIL do
      begin DazuKnotenListe(Liste,V^.nach);V:=V^.next end
  end;
procedure AlleendinListe(B:Knoten;var Liste:Verweis);
  var V:Verweis;                     (* Sammeln aller End-Knoten in    *)
  begin                              (* in einer Liste                 *)
    if B^.nL=NIL then DazuKnotenListe(Liste,B)
    else begin V:=B^.nL;
      while V<>NIL do begin AlleendinListe(V^.nach,Liste);V:=V^.next
  end end end;
procedure RueckinListe(P:Knoten;var Liste:verweis);
  begin Liste:=NIL;                  (* Sammeln aller Knoten von P auf-*)
    while P<>NIL do                  (* waerts in einer Liste          *)
      begin DazuKnotenListe(Liste,P);P:=P^.vor
  end end;
```

Bild 52 Prozeduren zur Baumanalyse

```
program Baumanalyse;

type Eintragtyp = record ident:string[20];zahl:integer end;

Function Gleich(E1,E2:Eintragtyp):boolean;
  begin if E1.ident=E2.ident then gleich:=true else gleich:=false end;

(*$I Baum.INC  *)    (* Operationen fuer beliebig verzw.Baeume *)

Procedure DruckeKnoten(P:Knoten);
  begin write(P^.Eintrag.ident,' ':20-length(P^.Eintrag.ident)) end;

(*$I KnotenLi.INC *)          (* Operationen fuer Knoten-Listen *)
(*$I Baumanal.INC *)          (* Operationen fuer Baum-Analyse  *)

var   Haeptop,P,B:Knoten;
      List:Verweis;E:Eintragtyp;
      wahl:char;

begin (* Hauptprogramm *)
  mark(Haeptop);B:=NIL;List:=NIL;
  write('Eingabe Kopfknoten:');readln(E.ident);
  NeuerAst(B,E);              (* Setzen Kopfknoten *)
  repeat
    write('Anfuegen an:');readln(E.ident);FindeKnoten(B,E,P);
    if E.ident<>'Ende' then
      if P <> NIL then repeat
          write('Anfuegen wen:');readln(E.ident);
          if E.ident<>'' then NeuerAst(P,E)
        until E.ident=''
      else begin
        writeln('Moegliche Anfuegeknoten sind:');Zweiginliste(B,List);
        DruckenknotenListe(List);LoeschenknotenListe(List) end
  until E.ident='Ende';     (* Einlesen beendet *)

  repeat                      (* Begin Auswertung *)
    write('Informationen zu:');readln(E.ident);
    Findeknoten(B,E,P);   (* P-gefundener Knoten *)
    if P<> NIL then begin
      write('(V)orgesetzte ! (a)lle/(d)irekte Unterstellte :');
      read(kbd,wahl);writeln(wahl);
      case wahl of
       'V','v':RueckinListe(P,List);
       'A','a':ZweiginListe(P,List);
       'D','d':UnterinListe(P,List) end;
      DruckenKnotenListe(List);writeln;
      LoeschenKnotenListe(List);end
    else write('Knoten mit Eintrag:  ,E.ident,' nicht enthalten');
  until E.ident='';
  release(Haeptop);
end.
```

Bild 53 Programm zur Baumanalyse

von der konkreten Typvereinbarung für den Typbezeichner *Eintragtyp* abhängig. Seine Vereinbarung muß verständlicherweise vor allen anderen Komponenten der Lösung erfolgen. Im Beispiel wird diesem Typ ein Record-Charakter vermittelt, wobei im Programm aber lediglich das Feld *ident* genutzt wird. Anhand dieses Feldes wird in der Funktion *gleich* auch die Entscheidung getroffen, ob zwei Knoten identisch sind. Gleichfalls wird auch nur dieses Feld in der Prozedur *DruckeKnoten* zur Ausgabe angewiesen. Diese Ausgabe auf dem Bildschirm erscheint jeweils linksbündig in einem 20stelligen Bereich. Der Anweisungsteil des Beispielprogrammes selbst kann in zwei Teile untergliedert werden. Im ersten Teil wird im Hauptspeicher der Baum aufgebaut und im zweiten Teil erfolgt die Auswertung der im Baum enthaltenen Informationen.

Der Aufbau des Baumes, der in der Praxis aus einer Datei heraus erfolgen wird, ist hier schrittweise programmiert. Zuerst wird nach dem Namen für den Kopfknoten gefragt und der Baum mit diesem Kopfknoten eröffnet. Die Variable B dient im ganzen folgendem Programm als Zeiger auf diesen Kopf des Baumes und wird nicht wieder verändert. Anschließend wird der Aufbau der einzelnen Verzweigungen organisiert. Dazu wird jeweils zuerst nach dem Knoten gefragt, an den neue Knoten angefügt werden sollen. Ist dieser Knoten vorhanden, so können an diesen beliebig viele Verzweigungen *angehängt* werden. Existiert der Knoten, an den angefügt werden soll, dagegen nicht, so wird der Bediener über alle bereits im Baum aufgenommenen Knoten informiert. (Nur an bereits aufgenommene Knoten des Baumes können weitere *angehängt* werden.)

Nachdem der Baum komplett aufgebaut wurde, der Nutzer die Frage nach dem nächsten Anfügeknoten also mit *Ende* beantwortet hat, beginnt die Auswertung der Baumstruktur. Dazu wird jeweils zuerst gefragt, zu welchem Knoten Informationen ermittelt werden sollen. Dieser Knoten wird im Baum gesucht, die Variable P dient dabei als Zeiger. Wenn der Knoten gefunden wurde, enthält dieser Zeiger nach Beendigung der Prozedur *FindeKnoten* einen von NIL verschiedenen Wert. In diesem Fall werden drei verschiedene Recherchen angeboten. Die vom Bediener gewünschte Analyse wird durchgeführt und unabhängig davon, welche es war, enthält die Variable *List* den Zeiger auf die Ergebnisliste der konkreten Recherche. Diese Liste muß jetzt nur noch durch die Prozedur *DruckeKnotenListe* ausgegeben werden. Nach Ausgabe der Liste auf den Bildschirm wird sie gelöscht, und die nächste Recherche zu einem neuen Knoten kann beginnen.

Dieses Beispielprogramm soll lediglich die Nutzung der entwickelten Komponenten verdeutlichen. Daher wurde es auch nur mit einem Minimum an Dialogführung programmiert. Gerade der Gestaltung des Nutzerdialogs muß in konkreten Programmen natürlich sehr viel mehr Gewicht beigemessen werden.

Zum Abschluß soll noch auf eine weitere mögliche Ausbaustufe der vorgestellten Baumstruktur hingewiesen werden. Bisher verwies ein Knoten auf mehrere Nachfolger, aber nur auf einen Vorgänger. Anstelle des direkten Vorgängerverweises ist aber auch eine Verweisliste für die Knotenstruktur möglich. So erhalten wir eine neue Qualität, denn jeder Knoten kann jetzt eine beliebige Anzahl von Vorgängern und Nachfolgern haben. Die Vielfalt solcher Datenstrukturen erweitert sich erheblich. Aus den Knotenbäumen werden Knotennetze (Bild 54). Die Vorstellung von Prozeduren und Funktionen zum Aufbau und zur Analyse solcher Knotennetze erübrigt sich. Sie können vom interessierten Leser leicht durch kleine Änderungen in den Listings der Bilder 50, 51 und 52 erarbeitet werden.

Texteditor

Nachdem wir uns in den Teilen 4 und 5 ausreichend mit den Pointern vertraut gemacht haben, werden wir sie in der im weiteren vorzustellenden Lösung, einem kleinen Texteditor, ob ihrer Effektivität des öfteren wiederfinden. Mit Editoren hat praktisch jeder zu tun, der irgendwie mit einem Rechner arbeitet. Die Anzahl der vorhandenen Editoren scheint fast unbegrenzt. Und trotzdem gibt es viele Programmierer, die selbst bereits mehrere Editoren geschrieben haben. Warum ist das so? Jeder Programmierer beginnt seine Arbeit am Rechner mit einem, meist rein zufällig ausgewählten, Editor. Diesen beginnt er zu beherrschen, und seine Grundvorstellung von einem Editor wird geprägt. Nun sind aber Software-Entwickler die kritischsten Nutzer von Software überhaupt, und dieser erste Editor wird irgendwann irgendwelchen Wünschen nicht mehr genügen. Es beginnt die Suche nach dem optimalen Editor, den es aber offensichtlich nicht gibt. Das vorläufige Ende dieser Suche bei einem Programmierer ist eigentlich selbstverständlich: Er schreibt sich seinen eigenen Texteditor. Und dieser realisiert dann meist den Grundumfang des zuerst benutzten Editors, erweitert um die speziellen Wünsche des jeweiligen Software-Entwicklers.

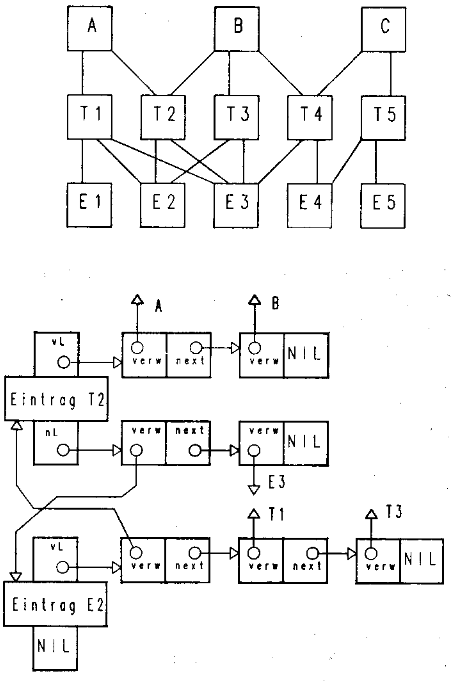


Bild 54 Vernetzte Knoten

Es gibt aber noch einen zweiten Grund, sich selbst einen Editor zu schreiben. Man braucht ihn als Quelltext, um ihn in eigene Programme einbinden zu können. Da dieses Problem nicht neu ist, existieren selbstverständlich Tool-Boxen zu diesem Thema. Aber diese meist erst für 16-Bit-Rechner angebotenen Lösungen haben oft eine Quelltextgröße von mehreren tausend Zeilen. Dieser Umfang resultiert meiner Meinung nach aus einem unnötigen Perfektionismus. Es erscheint daher einfacher, einen eigenen, kleinen Editor zu schreiben, als einen dieser *Riesen* zu kürzen und für eine Nutzung auf 8-Bit-Technik umzuschreiben.

Nun zur konkreten Lösung. Als erstes gilt es zu definieren, was ein Text ist, und wie er im Programm behandelt werden soll. In der im folgenden vorgestellten Lösung werden die Zeilen in Form einer dynamischen Listenstruktur gehalten. Die Zeilen stellen dabei ihrerseits Turbo-Pascal-Strings dar. Die maximale Länge der einzelnen Zeilen ist dabei für den gesamten Text gleich, und auf 100 Zeichen begrenzt. Die Anzahl der Zeilen, oder auch die Länge der Textzeilenliste, wollen wir durch den Hauptspeicher begrenzt lassen. Das heißt, ein Retten von Textteilen während des Editierens auf ein Massenspeichermedium ist in dieser einfachen Lösung nicht vorgesehen. Damit ist die Nutzbarkeit auf relativ kleine Texte eingeschränkt.

Die Unterscheidung und Identifikation der Texte erfolgt stets durch einen Zeiger auf ihre erste Zeile, das heißt auf das jeweils erste Element der Textzeilenliste. Auf Grundlage dieser Definition ist es dann auch möglich, mehrere Texte gleichzeitig im Hauptspeicher zu halten und wechselseitig mit den gleichen Funktionen oder Prozeduren zu behandeln. Um dies zu ermöglichen, wird jeder Teilkomponente der Lösung stets über einen Variablenparameter dieser Textkopf übermittelt.

Die einzelnen Komponenten der Lösung sind zur Erläuterung in einen Programmrahmen zusammengefaßt worden (Bild 56). Die Typvereinbarungen am Beginn des Programmes definieren die oben besprochene und zu behandelnde Listenstruktur. Die Variable Textkopf wird als Zeiger auf diese Liste genutzt. Die Anpassung an die konkrete Hardware wird, wie bereits des öfteren beschrieben, über ein entsprechendes Include-File (Bild 55) organisiert, das die benötigten Bildschirmsteuerzeichen und Tastencodierungen beinhaltet. Sie werden im weiteren vorausgesetzt. Weiterhin sind diesmal auch die Eingabeprozedur getkey und die Ausgabe writexy ausgelagert worden, um eine leichtere Anpassung an eine andere Hardware vorzubereiten. Die Funktion *inssatz* und die Prozedur *delsatz* sind für den satzweisen Auf- bzw. Abbau der Textliste im Hauptspeicher verantwortlich. Beiden Konstruktionen wird über den ersten Parameter mitgeteilt, mit welcher Liste sie arbeiten soll, da ja die Arbeit mit mehreren Listen möglich sein soll. Die Prozedur delsatz erhält im zweiten Parameter den Zeiger auf den zu löschenden Satz in der Liste. Der Funktion inssatz wird über die Parameter *vor* und *nach* eindeutig mitgeteilt, an welcher Stelle der Textliste ein neuer Satz eingefügt werden soll. Als Ergebnis von inssatz wird ein Zeiger auf den neuen Satz zurückgegeben.

Es folgen zwei Komponenten, die die Verbindung von den Textdateien auf der Diskette zu der Textliste im Hauptspeicher herstellen. Die Prozedur *Textsichern* ist in der Lage, einen Teil einer Textliste, der durch die Parameter *von* und *bis* eindeutig bestimmt ist, in eine Diskettendatei abzuspeichern. Soll die komplette Liste abgespeichert werden, so muß *von* auf den Textkopf zeigen und in bis muß NIL eingetragen werden. Die Funktion *Textlesen* gestattet die Umkehrung dieses Vorganges, und ist in der Lage, in eine bereits existierende Liste eine Diskettendatei einzufügen. Die genaue Stelle in der Liste ist durch die Parameter *vor* und *nach* bestimmt. Soll an den Anfang einer Liste herangelesen werden, so ist in vor der Zeigerwert NIL und in nach der Textkopf einzutragen. Für den Neuaufbau einer Textliste ist sowohl in vor wie auch in nach ein NIL anzugeben. Die Variable Textkopf wird von der Funktion in jedem Fall berichtigt zurückgegeben. Zur Bestimmung der Bildschirmzeile, in der die Fragen nach den Dateinamen erfolgen sollen, wird beiden Komponenten über y eine Zeilennummer übergeben.

Nun folgt mit der Funktion *Textedit* der Editor selbst. Ihm werden drei Parameter übergeben. Die ersten beiden bestimmen den Bildschirmbereich, in dem der Editiervorgang er-

```
   (* Vereinbarungen fuer  PC 1715   unter CPA  *)
const
  Nop         =   #00;    (*  Quasileeres Char     *)
  Enter       =   #13;    (*  Enter-Taste          *)
  Escape      =   #27;    (*  Escape-Taste         *)
  kursortief  =   #24;    (*  Kursortaste tief     *)
  kursorhoch  =   #05;    (*  Kursortaste hoch     *)
  kursorrechs=    #04;    (*  Kursortaste rechts   *)
  kursorlinks=    #08;    (*  Kursortaste links    *)
  pageup      =  #$12;    (*  Page up              *)
  pagedown    =  #$03;    (*  Page down            *)
  tablinks    =  #$01;    (*  Wort links           *)
  tabrechs    =  #$06;    (*  Wort rechts          *)
  DEL         =  #$7F;    (*  DEL - Taste          *)
  CE          =   #24;    (*  CE  - Taste          *)
  STaste      =    #2;    (*  S   - Taste          *)
  kursoron    =  #$82;    (*  Kursor ein           *)
  kursoroff   =  #$83;    (*  Kursor aus           *)
  normvideo   =  #$84;    (*  normale Darstellung  *)
  invsvideo   =  #$85;    (*  inverse Darstellung  *)
  intsvideo   =  #$86;    (*  intensive Darstellung *)
  ivitvideo   =  #$87;    (*  invers und intensiv  *)
  delrest     =  #$16;(* Zeile ab Position loeschen *)
  delline     =  #$18;(* ganze Zeile loeschen       *)
  clrscrc     =  #$0C;(* Loeschen ganzen Bildschirm *)
  cleardown   =  #$14;(* Loeschen ab Kursorposition *)

procedure getkey(var c:char);
   begin  read(kbd,c)    end;
procedure writexy(x,y:byte;s:string80);
   begin  gotoxy(x,y);   write(s)  end;
```

Bild 55 Vereinbarungen für den PC 1715

```
program editor;
type string80  = string[80];
     zeilentyp = string[100];
     pointer   =^pointr;
     pointr    = record s:zeilentyp;v,n:pointer end;
var  Textkopf,Heaptop:pointer;

{$I cons1715.inc}

function exist(s:string80):boolean;
  var d:file;
  begin assign(d,s);{$I-};reset(d);{$I+};
    if ioresult<>0 then exist:=false else exist:=true;
    close(d)
  end;

procedure delsatz(var Textkopf:pointer;aktu:pointer);
  begin
    if aktu^.v<>NIL then aktu^.v^.n:=aktu^.n
                    else Textkopf  :=aktu^.n;
    if aktu^.n  <>NIL then aktu^.n^.v:=aktu^.v;
    dispose(aktu) end;
function
inssatz(var Textkopf:pointer;vor,nach:pointer):pointer;
  var tem:pointer;
  begin
    if ((maxavail<85) and (maxavail>0))  then
      begin inssatz:=NIL;exit end;
    new(tem);tem^.n:=nach;tem^.v:=vor;tem^.s:='';
    if vor<>NIL  then  vor^.n:=tem else Textkopf:=tem;
    if nach<>NIL then nach^.v:=tem;
    inssatz:=tem end;

procedure Textsichern(y:byte;von,bis:pointer);
  var s,Dateiname:string80;
      d:text;
  begin if von<>NIL then begin
    writexy(55,y,'Datei-Sichern:'); read(Dateiname);
    if Dateiname<>'' then begin
      assign(d,Dateiname); rewrite(d);
      while((von<>NIL) and (von<>bis)) do
        begin writeln(d,von^.s); von:=von^.n end;
      close(d);
  end end end;

function Textlesen
  (y:byte;var Textkopf:pointer;vor,nach:pointer):pointer;
  var Dateiname:string80;
      d:text;
  begin if vor=NIL then Textkopf:=NIL;
    writexy(55,y,'Datei-Laden  :'); read(Dateiname);
    if exist(Dateiname) then begin
      assign(d,Dateiname); reset(d);
      while (not(eof(d))and((MaxAvail>85)or(Maxavail<0)))do
        begin
          vor:=inssatz(Textkopf,vor,nach); readln(d,vor^.s)
        end;
      if not(eof(d))then {Datei nicht komplett eingelesen!};
      close(d) end;
    Textlesen:=Textkopf
  end;
```

```
function Textedit(y0,ye:byte;Textkopf:pointer):pointer;
  var x,y,i:byte;c:char;znr:integer;
      Tem,aktu    :pointer;
  procedure schr(y:byte;aktu:pointer);
    begin writexy(1,y,aktu^.s+delrest) end;
  procedure schreiben(y:byte;poi:pointer);
    begin while (poi<>NIL) and (y<=ye) do
      begin schr(y,poi); y:=y+1; poi:=poi^.n end;
      if y<=ye then writexy(1,y,delline);
    end;
  procedure bild(y:byte;poi:pointer);
    begin while (y<>y0) and (poi^.v<>NIL) do
      begin y:=y-1; poi:=poi^.v end;
      if (y<>y0) then writexy(1,y-1,delline);
      schreiben(y,poi)
    end;
  function worttest:boolean;
    const fest :set of char = ['A'..'Z','a'..'z','0'..'9'];
    begin
      worttest:=((aktu^.s[x]in fest)and not(aktu^.s[x-1]in fest))
    end;
  procedure zeilv; begin aktu:=aktu^.v ;znr:=znr-1 end;
  procedure zeilp; begin aktu:=aktu^.n ;znr:=znr+1 end;

  begin     (*  Begin des  Hauptblockes der Funktion Textedit *)
    write(kursoroff);
    if (Textkopf=NIL) then Textkopf:= inssatz(Textkopf,NIL,NIL);
    aktu:=Textkopf; znr:=1; x:=1; y:=y0; c:=#0;
    repeat for i:=y0 to ye+1 do writexy(1,i,delline);
     bild(y,aktu);
     repeat  gotoxy(35,ye+1);
       write(znr:4,' Spalte:',x:3,' DEZ=',integer(c):4,kursoron);
       gotoxy(x,y);getkey(c); write(kursoroff);
       case c of
        ^E,kursorhoch :if (aktu^.v<>NIL) then begin zeilv;
                          if y>y0 then y:=y-1
                                  else schreiben(y,aktu) end;
        ^X,kursortief :if (aktu^.n<>NIL) then begin zeilp;
                        if y<ye then y:=y+1 else bild(y,aktu) end;
        ^D,kursorrechs:if x<80 then x:=x+1;
        ^S,kursorlinks:if x>1  then x:=x-1;
        ^A,Tablinks:if x>1 then
                       repeat x:=x-1 until worttest or(x=1)
                     else if (aktu^.v<>NIL) then begin zeilv;
                       x:=length(aktu^.v^.s)+1;
                     if y>y0 then y:=y-1 else schreiben(y,aktu)end;
        ^F,Tabrechs:if (x<=length(aktu^.s)) then repeat
                      x:=x+1 until worttest or (x>length(aktu^.s))
                     else if aktu^.n<>NIL then begin zeilp; x:=1;
                      if y<ye then y:=y+1  else bild(y,aktu)  end;
        pageup  :begin for i:=y0 to ye-1 do begin
                   if aktu^.v<>NIL then begin zeilv;
                     if y>y0 then y:=y-1 end end;
                   if y=y0 then schreiben(y,aktu)  end;
        pagedown:begin for i:=y0 to ye-1 do begin
                   if aktu^.n<>NIL then begin
                     zeilp;if y<ye then y:=y+1 end end;
                   if y=ye then bild(y,aktu) end;
        ^W       :begin if ((y<ye) and (aktu^.n<>NIL)) then y:=y+1
                   else if (aktu^.v<>NIL) then zeilv;
                   bild(y,aktu) end;
        ^Z       :begin if ((y>y0) and (aktu^.v<>NIL)) then y:=y-1
                   else if (aktu^.n<>NIL) then zeilp;
                   bild(y,aktu) end;
        enter    :begin znr:=znr+1;
                    Aktu:=inssatz(Textkopf,Aktu,Aktu^.n);
                    if x=1 then begin
                      aktu^.s := aktu^.v^.s; aktu^.v^.s:='' end
                    else begin
                      if (x)<length(aktu^.v^.s) then begin
                        Aktu^.s   := copy(Aktu^.v^.s,x,78);
                        Aktu^.v^.s:= copy(Aktu^.v^.s,1,x-1) end;
                      x:=1 end;
                    if y<ye then begin y:=y+1;
                      schreiben(y-1,Aktu^.v) end
                    else bild(y,Aktu) end;
        DEL      :if x>1 then begin
                    x:=x-1; delete(aktu^.s,x,1);schr(y,aktu) end
                  else if Aktu^.v<>NIL then begin
                    x:=length(Aktu^.v^.s)+1;
                    Aktu^.v^.s := Aktu^.v^.s + Aktu^.s;
                    Tem:=Aktu; zeilv; delsatz(Textkopf,Tem);
                    if y>y0 then y:=y-1;schreiben(y,Aktu) end;
        ^G       :begin delete(aktu^.s,x,1); schr(y,aktu) end;
        ^Y       :begin x:=1;
                    if aktu^.n<>NIL then begin
                      aktu:=aktu^.n; delsatz(Textkopf,aktu^.v);
                      schreiben(y,aktu) end
                    else if aktu^.v<>NIL then begin
                          aktu:=aktu^.v;delsatz(Textkopf,aktu^.n);
                          bild(y,aktu) end
                         else begin aktu^.s:=''; schr(y,aktu) end
                  end;
        ^N       :begin x:=1;aktu:=inssatz(Textkopf,aktu^.v,aktu);
                    schreiben(y,aktu) end;
        ' '..#126:begin
                  while length(aktu^.s)<x do aktu^.s:=aktu^.s+' ';
                  insert(c,aktu^.s,x);schr(y,aktu);
                  if x<80 then x:=x+1 end;
      end; { case }
    until c in [^K,#27];
    writexy(1,ye+1,
'S-ichern R-Lesen D-Sichern&Ende Q-uit J-ump'+delrest+kursoron);
    getkey(c);c:=upcase(c);
    case c of
     'D','S',^D,^S: Textsichern(ye+1,Textkopf,NIL);
     'J'   : begin writexy(65,ye+1,'Jump-to:');readln(i);
               while (i<>znr)and(i<>0) do
                 if (znr<i) then
                   if aktu^.n<>NIL then zeilp else i:=0
                 else if aktu^.v<>NIL then zeilv else i:=0
             end;
     'R'   : Textkopf:=Textlesen(ye+1,Textkopf,aktu,aktu^.n) end;
   until c in ['Q','D','X',^Q,^D,^X,#27];
   Textedit:=Textkopf;
  end; { Textedit }

begin { Hauptprogramm }
  mark(Heaptop);clrscr;
  Textkopf:=Textlesen(24,Textkopf,NIL,NIL);
  Textkopf:=Textedit (1,23,Textkopf);
  release(Heaptop);
end.
```

Bild 56 Texteditor

folgen soll. Über den Parameter Textkopf wird, wie bereits beschrieben, die Textzeilenliste übergeben. Soll ein neuer Text eingegeben werden, muß hier NIL stehen. Der Ergebniswert der Funktion Textedit ist ein Zeiger auf die geänderte Liste.

Die lokalen Variablen der Funktion Textedit werden wie folgt genutzt: X und y sind die jeweils aktuelle Bildschirmposition des Kursors, x bestimmt damit auch die aktuelle Änderungspostion im jeweiligen String. Die Zeichenvariable c dient der Tastaturabfrage und damit der Funktionssteuerung. In der Integerzahl *znr* wird die aktuelle Zeilennummer mitgezählt. Mit der Pointer-Variablen *aktu* wird das jeweils in Arbeit stehende Textlistenelement identifiziert. *Tem* ist eine temporär genutzte Variable. Einige wenige Teilaufgaben sind als lokale Prozeduren oder Funktionen programmiert. Sie haben folgende Aufgaben: Die Prozedur *schr* schreibt die Zeile des mit aktu identifizierten Listenelementes auf die mit y bestimmte Zeile des Bildschirmes und löscht den Rest der Zeile. Der Prozedur *schreiben* wird die aktuelle Bildschirmzeile und das aktuelle Listenelement übergeben. Von dieser Zeile an aktualisiert die Prozedur das Bildschirmaussehen anhand der Textliste. Die Prozedur *bild* erhält als Parameter die gleichen Informationen. Sie aktualisiert aber den gesamten Bildschrim. Dem wortweisen Springen beim Editieren dient die Funktion *worttest*. Durch die Prozeduren *zeilv* und *zeilp* im Funktionskörper wird die starre Kopplung zwischen der zu bearbeitenden Textliste und dem Zeilenzähler *znr* generell garantiert.

Im Anweisungsteil der Funktion wird nun präzise definiert, welche Tastatureingabe welche Reaktion des Editors hervorruft. Die schwierigste Aufgabe dabei ist die ständige Übereinstimmung von Bildschirmaussehen und Textinhalt. Das eigentliche Editieren erfolgt in der inneren Repeat-until-Schleife, die fast nur aus einer Case-Verzweigung besteht. Die äußere Schleife beinhaltet zusätzliche komplexere Funktionen und kann noch viel weiter ausgebaut werden. Das hier organisierte Abspeichern (∧KD) und Lesen (∧KR) von Diskettendateien ist das notwendige Minimum. Das zusätzlich aufgenommene Springen (∧KJ) auf eine konkrete Zeile soll die Möglichkeit von Erweiterungen nur andeuten.

Schluß

✉ KONTAKT ☎

Manfred Zander, Schäferstraße 11, Dresden, 8010

Systeme mit System

Teil 3

Scheibenbremsen nicht gefragt

PC-Harddisk-Systeme

Michael Wilde

Preis und Leistung werden von der Speicherkapazität der Platte und der Geschwindigkeit des Gesamtsystems bestimmt, das aus Platte, Controller und Rechner besteht. Man sollte seine Bedürfnisse anhand der geplanten Anwendungen möglichst exakt ermitteln, denn unbedachte Überdimensionierung kann den Preis sprungartig hochschnellen lassen, während übertriebene Genügsamkeit die restlichen Komponenten des Rechners behindert.

Platz muß sein

Weniger als 20 MByte Plattenkapazität sind selbst in einfachen XTs heute nicht mehr sinnvoll, auch wenn der Aufstieg von einem 360-KByte-Diskettenlaufwerk auf eine gebrauchte 5- oder 10-MByte-Platte anfangs begeistern mag. Die Praxis zeigt aber, daß man beim Umstieg auf Festplatte eine neue Arbeitsweise entwickelt. Innerhalb kurzer Zeit benutzt man Disketten nur noch zum Datentransfer und als Backup-Medium. Auch mit Selbstdisziplin speichert man viel mehr auf der Platte, als man eigentlich im ständigen Zugriff haben muß.

So bestimmt letztlich die Gesamtheit der Anwendungen, die regelmäßig auf dem Rechner laufen sollen, den realen Bedarf an Plattenkapazität. Beispielweise stößt ein reines Textverarbeitungssystem auf einem XT nicht so bald an die Grenzen einer 20-MByte-Platte. Handelt es sich jedoch um ein universell eingesetztes Gerät, auf dem auch längere Texte geschrieben und Programme entwickelt werden sollen, so kann selbst der Hobby-Anwender schon mal die Meldung „Disk full" bekommen.

Jedes „bessere" Anwenderprogramm und sogar Spiele schlucken heutzutage gleich einige MByte. Installationsprogramme laden Mengen an Drucker-, Bildschirm-, Maus- und Tastaturtreibern auf die Platte, obwohl nur je einer gebraucht wird. Wen kümmert schon, daß DOS auch einer 10 Byte langen Datei 2KByte Plattenspeicher zuweisen muß, weil das die kleinste Verwaltungseinheit (Cluster) ist.

Programme wie Datenbanken sammeln stetig wachsende Datenbestände an. Es ist ratsam, sich vor der Plattenanschaffung zu informieren, wieviel Kapazität eine anvisierte Anwendung für einen geplanten Anwendungsfall auf längere Sicht erfordert. Zum einen kann man bei „Plattenüberlauf" nicht einfach Daten auf Disk auslagern, zum anderen bekommt man Plattenkapazität nur in Sprüngen von etwa 20, 40, 80 MByte bei MFM- und 30, 60, 90 MByte bei RLL-Platten, und die Preise machen diese Sprünge ungefähr mit.

Geschwindigkeit

Für den Anwender zählen nicht irgendwelche Spitzenwerte, die die Plattenhersteller auf dem Prüfstand ermittelt haben. Sein System aus Rechner, Controller und Platte bestimmt als Ganzes, wie schnell er Daten speichern und zurückholen kann. Die von Plattenherstellern gern herausgestellte „mittlere Zugriffszeit" sagt zum Beispiel nur aus, wie schnell sich der Kopfträger einer Platte bei Spurwechseln über die Platte bewegt. Je schneller das geht, desto besser ist das fraglos, es gibt aber bedeutend wichtigere Parameter.

Eine der wichtigsten Größen ist die Datentransferrate, deren Grenze von einfachen physikalischen Gesetzmäßigkeiten bei der Datenaufzeichnung vorgegeben wird. Die Festplatte besteht aus mehreren magnetisierbaren Scheiben, auf denen die gespeicherten Daten in kreisförmigen Spuren lagern. Eine Spur ist beim PC in mehrere Sektoren von 512 Byte Länge unterteilt.

Aufzeichnungsverfahren

Die beiden gebräuchlichsten Aufzeichnungsverfahren sind MFM (Modified Frequency Modulation) und RLL (Run Length Limited). RLL ermöglicht eine höhere Datendichte als MFM, es passen folglich mehr Sektoren in eine Spur. Zu MFM-Platten gibt es vom selben Hersteller fast immer baugleiche RLL-Versionen mit Plattenoberflächen höherer Güte. Während man unbesorgt eine RLL-Platte auch an einem MFM-Controller betreiben kann (unter Verlust der Vorteile, die die RLL-Datendichte bringt), beschert der Betrieb von MFM-Platten an RLL-Controllern meist nach kurzer Zeit eine Fülle defekter Sektoren.

Die maximale Datenrate läßt sich aus der Sektorzahl (17 bei MFM, meist 26 bei RLL), der konstanten Drehzahl (60 Umdrehungen pro Sekunde) und den Bytes pro Sektor ermitteln. Bei MFM-Platten erreicht man folglich maximal 522, bei RLL-Typen 799 KByte/s. Moderne Systeme wie SCSI- und ESDI-Platten arbeiten intern mit einer noch höheren Datendichte (meist Advanced-RLL, auch ARLL), wodurch sie weit über 30 Sektoren in der Spur unterbringen. Dementsprechend steigt die maximale Datenrate, sofern die Drehzahl der Platte nicht auf 50 Umdrehungen pro Sekunden verringert wird.

Interleave

Selbst ein schneller 80386-AT schafft es nicht, alle Sektoren einer Spur in nur einer Plattenumdrehung zu lesen. Nachdem das DOS einen Sektor vom Controller entgegengenommen und richtig im Speicher plaziert hat, ist beim Anfordern des Folgesektors dessen Anfang gerade unter dem Lesekopf vorbeigerauscht und erst nach einer vollen Plattenumdrehung wieder zugänglich. Damit sinkt die optimale Datenrate auf einen Sektor pro Umdrehung, als ein 17tel oder 26stel der oben angeführten Werte.

Hätte man im obigen Beispiel verlangt, die Daten in der Reihenfolge Sektor 1, Sektor 3, Sektor 5 zu lesen, dann wäre man nach zwei Plattenumdrehungen fertig. Genau dieser Trick, genannt Interleaving oder Sektorversatz, wird auch angewandt, wobei die Sektornumerierung schon auf der Platte durch die physikalische Formatierung geändert wird, im Beispiel also auf einen Interleave-Faktor von 2, mit dem auch 80286-ATs an MFM-Controllern zurechtkommen. Langsamere Rechner (XTs) brauchen einen größeren Interleave-Faktor (XTs etwa 4 bis 5), weil die Verwaltung durch DOS länger dauert.

Dementsprechend sinkt auch die Datentransferrate bei Interleave 5 auf ein Fünftel des Maximalwertes. Macht man den Interleave-Faktor jedoch zu klein, wird der Folgesektor stets verpaßt, und schlagartig hat man den höchstmöglichen Interleave und die geringstmögliche Datentransferrate eingestellt.

Ein Puffer im Harddisk-Controller, der mehrere Folgesektoren zwischenspeichert, lokkert die starre Kopplung zwischen zeitlicher Sektorabfolge und Zugriffsmöglichkeit seitens des Rechners. Der Controller füllt permanent den Puffer mit Folgesektoren, die er dann auf Anfrage sofort parat hat. Mit solchen Controllern kann man gefahrlos 1:1 wählen (sie werden deshalb 1:1-Controller genannt) und kommt damit auch recht dicht an die theoretischen Maximalwerte heran.

1:1-Controller sind nur unwesentlich teurer als die ungepufferten Kollegen, mit denen man nicht über einen Interleave von zwei hinauskommt. Der Preis für die dreifache Leistung (RLL-1:1 statt MFM 1:2) beträgt aber bei weitem nicht das Dreifache, sondern oft nur 10 bis 20 Prozent mehr. Achten Sie aber darauf, daß die Software zum Formatieren mit diesem Interleave dabei ist, wenn der Controller kein ROM-BIOS mit internem Formatter hat.

Für XTs gibt es unseres Wissens keine vollwertigen 1:1-Controller (die dürften XTs auch nicht ausnutzen können), aber man erreicht zum Beispiel mit Omti-Controllern (einige gepufferte Sektoren) bei RLL etwa Raten von 300 KByte/s. Den jeweils kleinstmöglichen Interleave kann man durch physikalisches Neuformatieren und Messen der Datenrate selbst herausfinden, Programme wie SpinRite zum Beispiel machen das sogar, ohne die Daten dabei zu löschen.

BIOS-Fragen

Auf dem XT-Controller befindet sich in einem eigenen ROM-BIOS alles Wissen, das die CPU braucht, um damit eine Festplatte zu steuern. Ein Standard-AT-Controller mit ST412/506-Schnittstelle wird jedoch vom AT-BIOS direkt bedient, das dazu eine Liste von verschiedenen Festplattenparametern vorrätig hält – allerdings grundsätzlich keine für RLL-Festplatten mit 26 Sektoren.

Aus dem Grund haben alle AT-RLL-Controller doch ein eigenes BIOS, in dem meist außer einem Installationsteil zur Eingabe der Plattenparameter noch ein Formatierprogramm residiert. Seine Hauptaufgabe besteht darin, auf der Platte selbst deren Parameter zu notieren, um sie beim Kaltstart dem AT-BIOS als die gültigen Parameter unterzuschieben.

Manche Software mag aber Platten mit 26 Sektoren nicht, selbst wenn man die Plattenparametertabelle im BIOS-ROM umpatcht (Novell Netware zum Beispiel). Das FDISK von DOS, das AT-ROM-BIOS und einige ältere Kombi-Controller sind ferner aufgeschmissen, wenn eine Platte mehr als 1024 Zylinder hat. Das hat zur Folge, daß man solche Platten damit nicht vollständig ausnutzen kann, also Speicherplatz verschenkt. Pfiffige Controller (etwa von Data Technology) rechnen dann in einem Emulationsmodus einfach die Plattenparameter auf mehr Köpfe um, so daß selbst das AT-BIOS fest daran glaubt,

eine Platte habe nur 17 Sektoren oder weniger als 1024-Zylinder.
Ein XT-Controller im AT kann eine sehr schlechte Wahl sein: Ein Omti-RLL-Controller, der bei uns 300 KByte/s in einem 8-MHz-XT brachte, schaffte nur 200 KByte in einem 8-MHz-AT (1 Wait). Im XT ist der vom Controller-BIOS unterstützte DMA-Transfer die richtige Lösung, weil er weitaus schneller als über Prozessor-I/O-Befehle abläuft. Der AT-Kombi-Controller wird jedoch vom AT-BIOS über 16-Bit-I/O-Befehle wesentlich effizienter angesteuert. Die Gründe: Die DMA des AT wird zwar mit fast gleichem Takt wie im XT betrieben, verliert aber bei einem Wait-State ein Drittel Geschwindigkeit gegenüber dem XT. Auch wird der Code im EPROM des Controllers (8 Bit breit, Waits durch Umsetzung auf 16-Bit-Bus der CPU) extrem langsam ausgeführt.

Unterlagen

Achten Sie beim Kauf einer gebrauchten Platte darauf, daß Ihnen der Verkäufer ein Datenblatt oder zumindest die Parameter mitgibt, die Sie auf ATs für das Setup beziehungsweise auf XT-Controllern beim physikalischen Formatieren angeben müssen. Es gibt kein Testprogramm, das diese Parameter selbst ermitteln könnte. Schreiben Sie sich diese Parameter und die Nummer des Setup-Eintrags am besten aufs Plattengehäuse, falls Ihre CMOS-Uhr mal vergeßlich wird. Wenn Sie ein neues Komplettsystem erwerben, achten Sie darauf, daß Sie zu Platte und Controller vernünftige Unterlagen bekommen.

Schnittstellen

Platten und Controller (RLL und MFM) mit ST412/506-Schnittstelle sind zur Zeit im PC-Markt dominant und am preisgünstigsten. Gespanne aus ESDI-Controller und -Platte sind etwas schneller als RLL-1:1-Lösungen, allerdings wenig verbreitet.
SCSI bietet auf lange Sicht die schnellstmöglichen und vielseitigsten Systeme, auch wenn die Aufzeichnungsdichten heutiger Platten noch kaum Vorteile gegenüber ESDI-Varianten bieten können. Obwohl SCSI-Lösungen heute noch deutlich teurer als ESDI-Kombinationen sind, wird die jetzt schon spürbare Akzeptanz von SCSI die Verbreitung wachsen und damit die Preise bald purzeln lassen.
An der SCSI-Schnittstelle lassen sich theoretisch acht beliebige Geräte anschließen, also nicht nur Festplatten, sondern auch etwa Streamer, optische Platten oder Laserdrukker. Besonders wenn Sie sich für eine Platte mit hoher Speicherkapazität entschieden haben, bei der ein Backup über Disketten nur noch schlecht möglich ist, sollten Sie die Anschaffung eines Systems mit einem Universal-SCSI-Controller in Erwägung ziehen. Solche gibt es beispielsweise von Future Domain, Western Digital und Adaptec, preislich liegen sie um 1 000 DM. Der Low-Cost-SCSI-Adapter von Seagate (unter 100 DM) unterstützt hingegen nur zwei Platten und auch nur solche aus dem eigenen Hause.
Einige Comupterhersteller (zum Beispiel Compaq, Commodore, Zenith) rüsten ihre Rechner mit hausgemachten Schnittstellen aus. Diese haben eine gewisse Ähnlichkeit mit SCSI, das heißt, das Interface auf der Rechnerseite ist sehr einfach (und preiswert), die hauptsächliche Controller-Elektronik liegt auf der Platte. Gegen eine solche Lösung wäre nichts einzuwenden, wenn Schnittstelle und Ansteuerung einheitlich wären. Zur Zeit jedoch ist man auf einige wenige Plattentypen – wenn nicht gar völlig auf teure Spezialversionen des Computerherstellers – festgenagelt.
Prüfen Sie in solchen Fällen genau das Angebot möglicher Platten und deren Preise. Es kann (vor allem bei späteren Erweiterungen) günstiger sein, ein Plattensystem mit Standard-Schnittstelle separat zu erwerben. Dazu müssen Sie aber vor dem Kauf definitiv klären, daß sich der interne Harddisk-Controller entfernen oder zumindest abschalten läßt.

Aufrüsten

Die Harddisk-Controller lassen fast immer den Anschluß einer zweiten Platte zu, wobei die neue Platte technisch (MFM, RLL, SCSI, ESDI oder „Hausmarke") definitiv zum Controller passen muß. Jedoch hängt es vom Gehäuse und der Bauhöhe der ersten Platte ab, ob dafür noch Raum ist. Die meisten (aber nicht alle) Gehäuse lassen den Einbau von vier Laufwerken (zwei Harddisks, zwei Floppies) in halber Bauhöhe zu, eine Harddisk in voller Bauhöhe verringert diese Anzahl auf drei.
In einem AT mit 20-MB-Platte und MFM-1:2-Controller sollte man sich überlegen, ob bei größeren Aufrüstungen (plus 40 MByte) nicht der Verkauf der alten Kombination aus Platte und Controller und die Anschaffung einer RLL-1:1-Kombination der reizvollere Weg ist. Diese Lösung ist etwa dreimal schneller, läßt noch Raum für eine weitere Platte und belastet das Netzteil geringer.
Der Parallelbetrieb zweier gleichartiger Controller in einem Rechner ist nicht möglich, doch arbeiten zum Beispiel einige Standard- und SCSI-Controller nebeneinander. Eine solche Zusammenarbeit ist aber keineswegs sicher und sollte vor einem Kauf intensiv erprobt werden.
3,5-Zoll-Platten werden im unteren Kapazitätsbereich (bis 200 MByte) über kurz oder lang die Typen mit 5,25-Zoll-Formfaktor völlig verdrängen. Vor allem die neuesten Ausführungen sind dabei deutlich leiser als ihre größeren Geschwister. Platten haben vom Hersteller spezifizierte Einbaulagen, das heißt, man darf sie beispielsweise nicht „auf dem Kopf" montieren, weil dabei Lagerschäden entstehen können. (gr)

/1/ D. Grell, Sektorroulette, Optimaler Interleave-Faktor bei Festplatten, c't 2/89, S. 128

Nervensägen kontra Laser-Show

Drucker von Typenrad bis Laser

Eckart Steffens

Die Qualität und Geschwindigkeit heutiger Matrixdrucker ist bereits im Bereich um 400 DM so hoch, daß man sie nicht gegen ein Profi-Modell tauschen würde, das vor fünf Jahren 2 000 DM gekostet hat. Dennoch gibt es auch heute noch deutliche Unterschiede in Preis, Leistung, Geräuschentwicklung, Einsatzeignung und Betriebskosten.

Zeichen wie gedruckt

Der Typenraddrucker wurde direkt von der Schreibmaschine abgeleitet; ein Schriftzeichen auf dem Typenrad wird direkt auf das Papier geschlagen. Der Druck erfolgt zeilenweise, bei relativ niedriger Druckgeschwindigkeit – etwa 25 Zeichen pro Sekunde. Typenraddrucker sind echte Schönschriftdrukker, bei denen sich mit nur wenigen Handgriffen das Typenrad gegen ein anderes austauschen und damit die Schrifttype wechseln läßt.
Wie Typenraddrucker gehören auch Matrixdrucker zur Gruppe der sogenannten Impact-Drucker, bei denen Zeichen durch mechanischen Anschlag zu Papier gebracht werden. Dadurch können sie Belege mit Durchschlägen drucken. Beim Matrixdrucker wird ein Zeichen jedoch nicht als Ganzes angeschlagen, sondern aus einer Pünktchenmatrix (zum Beispiel 5 × 9 Punkte) zusammengesetzt. Jeweils eine Pünktchenspalte wird dabei auf einmal gedruckt, für ein Zeichen benötigt der Drucker im Vorbeifahren daher maximal fünf Anschläge.
Damit muß er zwar mehr Arbeit leisten als ein Typenraddrucker, doch da eine erheblich kleinere Masse bewegt werden muß (die Drucknadeln sind sehr klein und leicht), kann der Anschlag auch sehr schnell erfolgen. Tatsächlich schaffen durchschnittliche Nadeldrucker bis zu 100 Zeichen/Sekunde, in der gehobenen Klasse 250 bis 300; Spitzenmodelle sind sogar mit 600 Zeichen/s und mehr zu bekommen. Die Qualität des Ausdruckes ist allerdings durchaus dürftig und eher für Tabellen und Kontrollausdrucke geeignet – mit dem Schriftbild der Typenradler kann die eines 9-Nadlers nicht konkurrieren. Damit das Schriftbild wenigstens für Briefe taugt, werden Drucker mit „Near Letter Quality" gebaut. Sie erzeugen durch einen doppelten, leicht versetzten Überdruck der Zeile ein wesentlich geschlosseneres Schriftbild, eben fast Schönschrift.

Je mehr, desto besser?

Um das Schriftbild weiter zu verbessern, verfeinerte man die Druckmatrix, indem man die Zahl der Drucknadeln im Druckkopf auf 24 erhöhte. Die Near Letter Quality, für die die 9-Nadler noch zwei Überdrucke benötigen, schaffen sie in einem Schnelldruckdurchgang, und bei verringerter Druckgeschwindigkeit verbessert sich die Ausgabequalität auf echte Briefdruckqualität. Mit Karbon- statt Textil-Farbbändern erreicht man eine noch bessere, gleichmäßig schwarze Deckung des Schriftbildes.
Nicht mehr nur zwischen Draft (Schnelldruck)

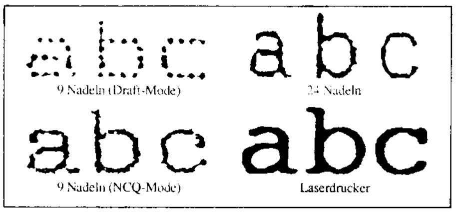
und LQ (Schönschrift) läßt sich bei heutigen Modellen wählen: Mehrere eingebaute Fonts (Schriftarten), die in einem EPROM geräteintern gespeichert sind und sowohl über ein Bedientableau als auch softwaremäßig angewählt werden können, sind selbst im Bereich unter 1 000 DM üblich. 48-Nadel-Drukker gibt es mittlerweile auch, jedoch sind sie noch in höheren Preisregionen angesiedelt.

Nicht klotzen, kleckern!

Matrixdrucker sind schnell, schönschriftfähig und eignen sich auch für Grafikausgabe. Doch gerade dabei ist die Lärmbelastung ganz erheblich. An Arbeitsplätzen, wo permanent ein solcher Drucker vor sich hin „sägt" (oder ein Typenradler hämmert), läßt sich das Betriebsklima nur mit Hilfe von Lärmschutzhauben heiter gestalten.
Ein Tintenstrahldrucker hat solche „Zusatzmaßnahmen" nicht nötig. Auch seine Zeichen werden aus einer Druckmatrix zusammengesetzt, wobei Tinte aus feinen Düsen (24, manchmal 48) auf das Papier gesprüht wird. Da kein Anschlag erfolgt und der Druckkopf das Papier nicht berührt, arbeitet der Drucker nahezu geräuschlos: Nur Papiervorschub und Druckkopfbewegung sind hörbar. Mit ihrer hohen Auflösung sind Tintenstrahldrucker echte Schönschriftler. Zugleich sind sie schnell und ihre Ausdrucke dokumentenecht, aber Durchschläge sind mit diesem Verfahren natürlich nicht zu erstellen.
Durchschläge ermöglichen auch die Thermodrucker nicht. Beim Thermodruck werden die einzelnen Druckkopfsegmente kurzeitig erhitzt, so daß sich das thermisch sensitive Papier an diesen Stellen verfärbt. Einige portable „Druckerschreibmaschinen", Tischrechner, Scanner-Kopierer und fast alle Telefaxgeräte arbeiten mit dieser Technik. Standardgeräte erreichen Auflösungen bis 200 dpi (Punkte pro inch), höhere Auflösungen sind erreichbar.
Thermotransferdrucker arbeiten ganz ähnlich, dabei wird aber ein spezielles Farbband benutzt, dessen erhitzte Punkte auch auf normalem Papier haften bleiben und zu sehr gleichmäßiger Deckung führen (wie Karbonband bei Typenradmaschinen). Ihre Druckqualität liegt daher über der von Tintenstrahldruckern.
Alle Arten von Thermodruckern sind Zeilendrucker und arbeiten sehr leise, verursachen jedoch ungleich höhere Druckkosten als andere Modelle, da entweder Spezialpapier (Thermopapier) oder Einmal-Thermofarbband benötigt wird.

Seite für Seite

Typenrad-, Matrix-, Tintenstrahl- und Thermodrucker verarbeiten Einzelblätter, Endlospapier, Etiketten und was sonst alles an Formularen angeboten wird. Der Einzug erfolgt mit der Walze (für Umschläge und Einzelblätter), über einen sogenannten Traktor (für die perforierte Führung bei Endlosformularen) oder einen Einzelblattschacht (für geschnittene Formate wie A4-Papier). Der Zubehörhandel hält vielerlei Ergänzungen für Sie bereit.
Bei Laserdruckern werden nur geschnittene Formate verarbeitet, allenfalls Umschläge können als „Einzelblatt" zugeführt werden. Auch die Drucktechnik unterscheidet sich von den vorgenannten Modellen beträchtlich: Hier wird nicht zeilenweise, sondern jeweils eine komplette Seite in einem Durchgang bedruckt – es kann also auch beliebig auf dieser



Etwas „dramatisiert" (nämlich aus Sicht eines Laserdrucker-Herstellers), aber anschaulich: die Unterschiede stark vergrößerter 10-Punkt-Schrift verschiedener Drucktypen.

Seite positioniert werden. Auch komplexe Grafiken sind so kein Problem, und mit der hohen Auflösung von 300 dpi erlauben Laserdrucker hochqualitative Ausdrucke. Laserdrucker sind dabei recht leise, aber auch sie müssen ohne Durchschlag auskommen. Neuere Modelle verwenden statt des Lasers auch eine LED-Zeile oder ein LCD-Shutter zum Belichten. Das hat auf das Druckergebnis keinen Einfluß, macht diese Modelle aber etwas preiswerter.

Schnittstelle und Emulation

Zwei Standard-Schnittstellen haben sich für den Druckeranschluß an den Rechner durchgesetzt: RS-232-C für die serielle Datenübermittlung, bei der parallelen Datenübermittlung der Vorschlag der Firma Centronics (weshalb man hier auch von der Centronics-Schnittstelle spricht).
Bezüglich Steuerkommandos, Schriftarten und so weiter jedoch gilt seit jeher: „Jeder Drucker ist anders". Softwarehersteller kompensieren das, indem sie zu ihren Produkten sogenannte Druckertreiber ausliefern, Anpassungsprogramme also, die die Ausgabe so formatieren, daß der angeschlossene Drucker sie versteht und das „richtige" Ergebnis zu Papier bringt.
Umgekehrt sind fast alle Drucker heute ihrerseits in der Lage, das Verhalten anderer Drucker nachzubilden, zu „emulieren", und so eventuelle Anpassungsprobleme im Hinblick auf verfügbare Treiber weiter zu verringern. Typenrad-Emulationsstandards wurden durch Qume- und Diablo-Geräte gesetzt, sie sind vor allem für Textverarbeitungszwecke von Bedeutung. Matrixdrucker-Standards setzten Epson mit der Druckersprache ESC/P und IBM. Die Laserdrucker-Standards gehen vor allem auf Hewlett Packard mit dem Laserjet und der HP Printer Control Language sowie Canon zurück, deren Druckwerke sich bei vielen Anbietern unter der Haube tummeln.
Ein anderes Problem, nur auf den zweiten Blick erkennbar, ist der Zeichensatz. So haben zum Beispiel Macintosh, IBM PC, Atari ST und Commodore Amiga für die Sonderzeichen oberhalb 128 (zu denen auch die deutschen Umlaute gehören) nicht nur zum Teil deutlich andere Codes, sondern auch einen völlig abweichenden Zeichenvorrat. In der Regel müssen Zeichensätze am Drucker unabhängig von der Emulation gesondert eingestellt werden. Mit IBM- und Epson-Zeichensatz, bei Laserdruckern Roman-8, klappt's in den meisten Fällen.

Welcher Drucker für wen?

Für gelegentliche Ausdrucke von Text und Grafiken bei einer Auflösung um 120 dpi und einer geringen Belastung des Budgets empfiehlt sich durchaus ein einfacher 9-Nadel-Matrixdrucker ab etwa 400 DM. Von Geräten in dieser Preisklasse sollte man keine Wunder an Langlebigkeit erwarten, bei einem mehrstündigen Non-Stop-Ausdruck schaltet auch schon mal die Übertemperatursicherung ab. Streßfähige Modelle für den rauhen Profi-Einsatz findet man oberhalb 1 000 DM. Ein Gerät, das normale A4-Formate, auch endlos, verarbeiten kann, ist im allgemeinen hinreichend. Ausführungen für Breitformate (A4 quer) gibt es zwar auch, allerdings selten im Low-Cost-Bereich.
Wenn keine längeren Listings zu drucken sind, dann eignet sich für gelegentliche Schönschreib-Exerzitien auch eine relativ langsame (15 Zeichen pro Sekunde) Typenrad-Schreibmaschine mit Computer-Interface (800 bis 1 800 DM). Gegenüber einem echten Typrenraddrucker hat sie sogar den Vorteil, daß man sie auch weiterhin per Hand bedienen kann, etwa um knifflige Formulare auszufüllen.
Als schnelle Schönschriftler mit vielen Fonts eignen sich vor allem 24-Nadel-Matrixdrukker (1 000 DM bis 3 000 DM), Tintenstrahl- oder Thermodrucker (1 500 bis 4 000 DM). Lediglich die Ausgabe einer Grafik ist mit diesen Geräten etwas lästig, weil die Vorlage ja zeilenweise zerlegt und ausgedruckt werden muß.
Seitendrucker wie Laser-, LCD- oder LED-Drucker haben es da leichter, sie können eine Seite auf einmal belichten. In der Grundausstattung sind diese Modelle meist aber nicht mit genügend Speicher versehen (mindestens 1,5 MByte), um eine komplette Grafikseite aufbauen zu können. Dennoch sind die zunehmend im Preis fallenden Seitendrucker (Grundgeräte sind bereits unter 2 500 DM erhältlich) eine starke Alternative zu allen anderen Modellen, denn sie sind sauber, leise, schnell und erzeugen in Verbindung mit leistungsfähiger Software durchaus druckfähige Vorlagen.
Laserdrucker werden aber schnell teuer (5 000 bis 15 000 DM), wenn man nach Versionen Ausschau hält, die eine Seitenbeschreibungssprache wie etwa Postscript beherrschen. Diese sind dann mit einem weitaus aufwendigeren Controller und minimal 3 oder 4 MByte Speicher ausgestattet, was man definitiv nur für professionelles Desktop Publishing braucht. Sucht man nur einen leisen Seitendrucker mit den Möglichkeiten eines guten Matrixdruckers, tut es ein ganz simples Basismodell.
Bei den laufenden Betriebskosten liegen Drucker, die mit Textilfarbbändern auskommen, am günstigsten, also Matrix und Typenrad. Dann folgen Tintenstrahldrucker, die mit einer Füllung für 65 DM etwa 3 Millionen Zeichen schaffen. Laserdrucker-Kartuschen für etwa 300 DM halten ungefähr 6 Millionen Zeichen durch. Führend sind Thermotransfer-Drucker: ein 30 bis 40 DM teures Einmal-Farbband reicht nur für 200 000 bis 300 000 Zeichen. (gr)

Schluß

Nachdruck aus c't – Magazin für Computertechnik, Heft 1/1990

Berechnung einer zur Run-Time eingegebenen Funktion

Peter Fölsche, Zerbst

Bei vielen Anwendungen erhöht es den Komfort und die Vielseitigkeit des Programms, wenn eine variabel zu berechnende Funktion im Dialog mit dem Programm eingegeben werden kann, anstatt sie in den Programmtext einbinden und neu compilieren zu müssen. Letztere Variante ist insbesondere bei Nutzern ohne Kenntnisse des Programms oder der verwendeten Sprache untauglich. Bisher wurde dieses Problem durch einen Interpreter gelöst. Das hatte die Nachteile eines höheren Softwareaufwandes und einer geringeren Rechengeschwindigkeit gegenüber der Berechnung durch einen Code, der durch einen Compiler erstellt wurde.

Ich möchte Ihnen im folgenden zwei Varianten demonstrieren, die *keinen* Interpreter benutzen, sondern den zu Turbo-Pascal 4.0/5.0 gehörenden Kommandozeilen-Compiler TPC.EXE. Mit der ab Turbo-Pascal 4.0 vorhandenen Prozedur

Exec (Path : String; ComLine : String)

kann ein Programm vom laufenden Programm aus wie eine Prozedur gestartet werden. Das heißt, nach Abarbeitung des aufgerufenen Programms läuft das aufrufende Programm mit dem nach *Exec* folgenden Befehl weiter. Damit ist es innerhalb eines laufenden Programms prinzipiell möglich, eine Funktion als String einzulesen, diesen String in ein vorbereitetes Pascal-Textfile *Name.pas* einzubinden, das Pascal-Textfile mit

Exec ('TPC.EXE', 'Name.pas')

zu compilieren und das entstandene Exe-File mit Exec aufzurufen. Bei Verwendung von Exec ist zu beachten, daß das aufrufende Programm nicht den gesamten Speicher belegt. Deshalb ist eine Begrenzung des Heap nach oben notwendig, das heißt, daß die Obergrenze des Heap mit der Compilerdirektive *Memory-Sizes* auf weniger als 640 KByte festgelegt werden muß.
Die hier angegebenen Programmbeispiele (Bild 1 bis 6) sind für Turbo-Pascal 5.0 gedacht, können aber schnell an Turbo-Pascal 4.0 angepaßt werden. Dazu sind die Compilerdirektiven neu zu setzen (einige gab es in Turbo-Pascal 4.0 noch nicht), die Prozedur SwapVectors und die Funktion

GetEnv (Str : String) : String;

sind selbst zu deklarieren (siehe Bild 7).

```
 1: { Unit zur Deklaration der Unterprogramme für 1. }
 2: {$A+,B-,D-,E+,F-,I-,L-,N+,O-,R-,S-,V-}
 3: Unit F1;
 4:
 5: interface
 6:
 7: uses Crt, DOS;
 8:
 9: type RealType = Extended;
10:
11: var ya          : array [0..767] of RealType;
12:     ey          : array [0..767] of Boolean;
13:     xd, x       : RealType;
14:     xi, xmax    : word;
15:     xa          : RealType;
16:
17: const Exist : Boolean = False;
18:
19: function Install( Fkt, Dkt : String ) : boolean;
20:
21: procedure Calculation;
22:
23: implementation
24:
25: var Path        : Text;
26:     SR          : SearchRec;
27:     TpcDir      : String;
28:     OldExitProc, Old47hPtr : Pointer;
29:
30: function Install;
31: var f1, f2 : Text; Zeile : String; DOSErrorCode : Integer;
32: begin
33: Assign( f1, 'calc.pas' ); Assign( f2, 'help1.pas' );
34: ReWrite( f1 ); ReSet( f2 );
35: while not EoF( f2 )
36: do begin ReadLn( f2, Zeile );
37:          if Pos( '{!Fkt}', Zeile ) <> 0
38:          then WriteLn( f1, Fkt )
39:          else if Pos( '{!Dkt}', Zeile ) <> 0
40:               then WriteLn( f1, Dkt )
41:               else WriteLn( f1, Zeile )
42:    end;
43: Close( f1 ); Close( f2 );
44: SwapVectors; Exec( TpcDir + 'Tpc.Exe', '/l calc.pas' ); SwapVectors;
45: WriteLn; DOSErrorCode:= DOSError;
46: if DOSErrorCode <> 0
47: then begin WriteLn( 'DOS-Fehler : ', DOSErrorCode ); halt
48:      end;
49: Exist:= DOSExitCode = 0; Install:= Exist
50: end;
51:
52: procedure Calculation;
53: begin
54: if not Exist
55: then begin Write( 'Es wurde noch keine Funktion installiert ! ');
56:            Write( 'Programm angehalten ! ' ); Halt
57:      end;
58: x:=xa; xi:=0; FillChar( ey, SizeOf( ey ), true );
59: repeat SwapVectors; Exec( 'calc.exe', '' ); SwapVectors
60:  until xi = succ( xmax )
61: end;
62:
63: {$F+} procedure NewExit; {$F-}
64: begin
65: ExitProc:= OldExitProc; SetIntVec( $47, Old47hPtr );
66: SwapVectors; Exec( GetEnv( 'COMSPEC' ) , '/c erase calc.*' );
67: SwapVectors
68: end;
69:
70: begin { Hauptprogramm }
71: DirectVideo:= False { nur bei EC 1834 / nicht kompatible };
72: Assign( Path, 'TpcDir.txt'); ReSet( Path );
73: Readln( Path, TpcDir ); FindFirst( TpcDir + 'Tpc.Exe', AnyFile, SR );
74: While DOSError <> 0
75: do begin WriteLn( 'Tpc.Exe not found !'); WriteLn;
76:          WriteLn( 'old Path : ', TPCDir ); WriteLn;
77:          Write( 'new Path : ' ); ReadLn( TPCDir ); WriteLn;
78:          FindFirst( TpcDir + 'Tpc.Exe', AnyFile, SR );
79:          ReWrite( Path ); Writeln( Path, TpcDir )
80:    end;
81: Close( Path ); OldExitProc:= ExitProc; ExitProc:= @ NewExit;
82: GetIntVec( $47, Old47hPtr ); SetIntVec( $47, @ ya )
83: end.
```

Bild 1 Die Unit F1.TPU zur Berechnung einer Funktion

```
 1: { zu 1. }
 2: { Hilfs-Programm zu  Unit 'F1.tpu' }
 3: {$A+,B-,D-,F-,I-,L-,N+,O-,R-,S-,V-}
 4: {$IfDef CPU87}  { Ist ein Numeric-Processor vorhanden ? }
 5: {$E-}           { ja => HardWare   }
 6: {$Else}
 7: {$E+}           { nein=> Emulieren }
 8: {$EndIf}
 9: {$M 65520,0,0 } { Es wird kein Heap benoetigt }
10: program help1;
11: uses Dos;
12: type RealType = Extended;
13:      Transfer = record ya         : array [0..767] of RealType;
14:                        ey         : array [0..767] of boolean;
15:                        xd, x      : RealType;
16:                        xi, xmax   : word
17:                 end;
18: var Back     : ^ Transfer;
19:     xn       : Word;
20:     OldExit  : Pointer;
21: {!Dkt}
22:
23: {$F+}
24: procedure NewExit;
25: begin
26: with Back^
27: do begin xi:=succ(xn); x:=x+xd; ey[xn]:=false
28:      end;
29: ExitProc:=OldExit; ErrorAddr:=nil
30: end; {$F-}
31:
32: function y( var x : RealType ) : RealType;
33: begin
34: {!Fkt}
35: end;
36:
37: begin
38: OldExit:=ExitProc; ExitProc:=@NewExit;
39: GetIntVec( $47, Pointer (Back) );
40: with Back^
41: do begin for xn:=xi to xmax
42:          do begin ya[ xn ]:= y( x ); x:=x+xd
43:             end;
44:          xi:=succ(xn)
45:    end;
46: ExitProc:=OldExit
47: end.
```

Bild 2 Hilfsprogramm zur Unit F1.TPU

```
 1: { zu 1. }
 2: { Beispiel-Programm zur Nutzung der Routinen des Units 'F1.tpu' }
 3: {$A+,B-,D-,E+,F-,I-,L-,N+,O-,R-,S-,V-}
 4: {$M 65520,0,0}
 5: program Main1;
 6:
 7: uses Crt, F1;
 8:
 9: var fkt : String;
10:     xe  : RealType;
11:
12: begin
13: ClrScr;
14: WriteLn( 'Tabellierung einer Funktion' ); WriteLn;
15: WriteLn( 'Die Funktion darf Singularitaeten enthalten !' );
16: WriteLn;
17: Repeat Repeat Write( 'Der Funktions-String muß folgendes ');
18:               WriteLn( 'Format besitzen : y:= f(x)' ); WriteLn;
19:               Write( 'Funktions-String : '); ReadLn( fkt );
20:               WriteLn
21:         until Install( fkt, '' );
22:        Write( 'Anfang       : '); ReadLn( xa ); WriteLn;
23:        Write( 'Ende         : '); ReadLn( xe ); WriteLn;
24:        Write( 'Schrittweite : '); ReadLn( xd ); WriteLn;
25:        xmax:= Round( ( xe - xa ) / xd ); Calculation; x:= xa;
26:        for xi:= 0 to xmax
27:        do begin if ey[ xi ]
28:                 then WriteLn( 'Funktion(', x, ')= ', ya[ xi ] )
29:                 else WriteLn( 'Funktion(', x, ')= ', 'NAN' );
30:                 x:= x + xd
31:           end;
32:        WriteLn;
33:        WriteLn( 'Wiederholen ? <J>a / <N>ein' )
34:  until UpCase( ReadKey ) = 'N'
35: end.
```

Bild 3 Beispielprogramm zur Nutzung der Unit F1.TPU

Berechnung einer Funktion ohne Laufzeitfehler

Es sollen in einem x-Intervall die Funktionswerte f(x) bzw. Iterationswerte von äquidistanten x-Werten (z. B. für eine graphische Darstellung) berechnet werden. Dazu werden folgende Variablen benutzt (siehe Bild 1, Unit F1 Zeile 11 bis 15):

Variable	Typ	Funktion
ya	RealType	Vektor, nimmt die Funktionswerte auf
ey	**Boolean**	Vektor, ey[xi] = true besagt, daß die Berechnung des Funktionswertes mit dem Index xi möglich war
xi	**word**	Index von ya und ey
xmax	**word**	maximaler Index, führt zum Ende der Berechnung
x	**RealType**	aktueller x-Wert
xd	**RealType**	Schrittweite für x
xa	**RealType**	Anfangswert für x

Die Funktion Install (Bild 1, Zeile 30 bis 50) kopiert das Textfile Help1.pas in das noch nicht existierende Textfile Calc.pas und trägt dabei den eingegebenen Funktions-String (y wird darin ein Wert zugewiesen) und eventuell zusätzliche Deklarationen dort in Calc.pas ein, wo es in Help1.pas die Hilfs-Strings {**!Fkt**} und {**!Dkt**} findet (Zeile 21 und 34). Dann wird versucht, Calc.pas zu compilieren. Die Funktion Install gibt *true* zurück, wenn die Compilierung erfolgreich war; *false*, wenn ein Fehler auftrat. Die Compilierung läßt sich auf dem Bildschirm verfolgen, und eine eventuelle Fehlerstelle wird von TPC.EXE angezeigt. Nach der erfolgreichen Compilierung kann das erzeugte Programm Calc.exe gestartet werden. In Help1.pas ist der Typ *Transfer* definiert (Zeile 13). Er enthält die zu übergebenden Variablen, die in der gleichen Reihenfolge definiert sind, wie die zu übergebenden Variablen in der Unit F1 (Zeile 11 bis 15). *Back* ist als Pointer auf eine Variable vom Typ Transfert deklariert. Daraus folgt: Wenn **Back: = Addr(ya)** (ya aus dem Hauptprogramm) gesetzt wird, liegen die zu übergebenden Variablen sowohl in der Unit F1 wie auch in Calc.exe auf dem gleichen Speicherplatz und können somit übergeben werden. Die Übergabe der Adresse der Variablen ya erfolgt im Initalisierungsteil der Unit F1 über den Interrupt-Vektor $47, der nach /1/ von MS-DOS nicht belegt ist und von Nutzerprogrammen verwendet werden kann.

Die Prozedur *Calculation* (in F1) führt, indem sie Calc.exe aufruft, die Berechnung aus. Tritt während der Berechnung der Funktion in Calc.exe ein Laufzeitfehler auf, wird das Programm Calc.exe über seine Exit-Prozedur (in Help1.pas, Zeile 24) beendet. Die Exit-Prozedur sorgt dafür, daß alle Parameter so ein-

```
 1: { Unit zur Deklaration der Routinen für 2. }
 2: {$A+,B-,D-,E-,F-,I-,L-,N+,O-,R-,S-,V-}
 3: Unit F2;
 4:
 5: interface
 6:
 7: uses Crt, DOS;
 8:
 9: type RealType = Extended;
10:
11: var FuncAddr : Pointer;
12:
13: const Exist : Boolean = false;
14:
15: function Install( Fkt, Dkt : String ) : boolean;
16:
17: function y( x : RealType ) : RealType;
18: inline( $FF/ $1E/ FuncAddr ); { Call FAR [FuncAddr] }
19:
20: implementation
21:
22: var OldExit, Old47hPtr, Old48hPtr
23:            : Pointer;
24:     Path   : Text;
25:     SR     : SearchRec;
26:     TpcDir : String;
27:
28: {$F+}
29: function NoFunction( x : RealType ) : RealType; {$F-}
30: begin
31: Write( 'Es wurde noch keine Funktion installiert ! ' );
32: Write( 'Programm angehalten !' ); halt
33: end;
34:
35: procedure ReleaseMemory;
36: var Regs : Registers;
37: begin
38: if Exist then Intr( $48, Regs );
39: Exist:=false; FuncAddr:= @ NoFunction
40: end;
41:
42: {$F+}
43: procedure NewExit;  {$F-}
44: begin
45: ExitProc:= OldExit; ReleaseMemory;
46: SetIntVec( $47, Old47hPtr ); SetIntVec( $48, Old48hPtr );
47: SwapVectors; Exec( GetEnv( 'COMSPEC' ) , '/c erase calc.*' );
48: SwapVectors
49: end;
50:
51: function Install;
52: var f1, f2 : Text; Zeile : String; DOSErrorCode : Integer;
53: begin
54: ReleaseMemory;
55: Assign( f1, 'calc.pas' ); Assign( f2, 'help2.pas' );
56: ReWrite( f1 ); ReSet( f2 );
57: While not EoF( f2 )
58: do begin ReadLn( f2, Zeile );
59:          if Pos( '{!Fkt}', Zeile ) <> 0
60:          then WriteLn( f1, Fkt )
61:          else if Pos( '{!Dkt}', Zeile ) <> 0
62:               then WriteLn( f1, Dkt )
63:               else WriteLn( f1, Zeile )
64:     end;
65: Close( f1 ); Close( f2 );
66: SwapVectors; Exec( TpcDir + 'Tpc.Exe', '/l calc.pas' );
67: SwapVectors; WriteLn; DOSErrorCode:= DOSError;
68: if DOSErrorCode <> 0
69: then begin WriteLn( 'DOS-Fehler : ', DOSErrorCode ); halt
70:        end;
71: if DOSExitCode = 0
72: then begin SwapVectors; Exec( 'calc.exe', '' ); SwapVectors;
73:              Install:= true; GetIntVec( $47, FuncAddr );
74:              Exist:= true
75:        end
76: else Install:= false
77: end;
78:
79: begin { Initialisierung }
80: DirectVideo:= false { nur bei EC 1834 / nicht kompatible };
81: Assign( Path, 'TpcDir.txt'); ReSet( Path );
82: Readln( Path, TpcDir );
83: FindFirst( TpcDir + 'Tpc.Exe', AnyFile, SR );
84: While DOSError <> 0
85: do begin WriteLn( 'Tpc.Exe not found !' ); WriteLn;
86:          WriteLn( 'old Path : ', TPCDir ); WriteLn;
87:          Write( 'new Path : ' ); ReadLn( TPCDir ); WriteLn;
88:          FindFirst( TpcDir + 'Tpc.Exe', AnyFile, SR );
89:          ReWrite( Path ); WriteLn( Path, TpcDir )
90:     end;
91: Close( Path ); FuncAddr:= @ NoFunction;
92: GetIntVec( $47, Old47hPtr ); GetIntVec( $48, Old48hPtr );
93: OldExit:= ExitProc; ExitProc:= @ NewExit
94: end.
```

```
 1: { zu 2. }
 2: { Hilfs-Programm zu  Unit 'F2.tpu' }
 3: {$A+,B-,D-,F-,I-,L-,N+,O-,R-,S-,V-}
 4: {$IfDef CPU87}  { Ist ein Numeric-Processor vorhanden ? }
 5: {$E-} { ja => HardWare }
 6: {$Else}
 7: {$E+} { nein=> Emulieren }
 8: {$EndIf}
 9: {$M 1024,0,0 } { Es wird kein Heap benoetigt ! }
10:
11: program help2;
12: uses dos;
13: Type RealType = Extended;
14:
15: {$F+}
16: function y( x : RealType ) : RealType; {$F-}
17: {!Dkt}
18: begin
19: {!Fkt}
20: end;
21:
22: function DeallocateMemory( Segment : word ) : word;
23: var Regs : Registers;
24: begin
25: with Regs
26: do begin ah:=$49; es:=Segment; MSDOS( Regs );
27:          if ( flags and 1 ) <> 0
28:          then DeallocateMemory:=ax
29:          else DeallocateMemory:=0
30:     end
31: end;
32:
33: Procedure DeallocateProgram; Interrupt;
34: type WordPtr = ^ Word; var Regs : Registers;
35: begin
36: if DeallocateMemory( WordPtr( Ptr( PrefixSeg, $2C ) )^ ) <> 0
37: then Intr( $19, Regs );
38: if DeallocateMemory( PrefixSeg ) <> 0
39: then Intr( $19, Regs )
40: end;
41:
42: begin
43: SetIntVec( $47, @ y ); SetIntVec( $48, @ DeallocateProgram ); keep(0)
44: end.
```

Bild 5 Hilfsprogramm zur Unit F2.TPU

```
 1: { zu 2. }
 2: { Beispiel-Programm zur Nutzung der Routinen
 3:   des Units 'F2.tpu' }
 4: {$A+,B-,D-,E+,F-,I-,L-,N+,O-,R-,S-,V-}
 5: {$M 65520,0,0}
 6: program Main2;
 7:
 8: uses Crt, F2;
 9:
10: type RealType = Extended;
11:
12: var x, xa, xe, xd : RealType;
13:     fkt           : String;
14:
15: begin
16: ClrScr;
17: WriteLn( 'Tabellierung einer Funktion' ); WriteLn;
18: Repeat Repeat Write( 'Der Funktions-String muß folgendes ');
19:               WriteLn( 'Format besitzen : y:= f(x)' );
20:               WriteLn;
21:               Write( 'Funktions-String : '); ReadLn( fkt );
22:               WriteLn
23:         until Install( fkt, '' );
24:        Write( 'Anfang        : '); ReadLn( xa ); WriteLn;
25:        Write( 'Ende          : '); ReadLn( xe ); WriteLn;
26:        Write( 'Schrittweite : '); ReadLn( xd ); WriteLn;
27:        x:=xa;
28:        Repeat WriteLn( 'Funktion(', x, ')= ', y( x ) );
29:               x:= x + xd
30:         until x > xe;
31:        WriteLn;
32:        WriteLn( 'Wiederholen ? <J>a / <N>ein' )
33:  until UpCase( ReadKey ) = 'N'
34: end.
```

Bild 6 Beispielprogramm zur Nutzung der Unit F2.TPU

◀ ***Bild 4 Die Unit F2.TPU zur Installation einer Funktion***

```
 1: { Routinen die in Turbo Pascal 4.Ø noch nicht verfügbar sind }
 2: Uses Dos;
 3:
 4: procedure SwapVectors;
 5: var Ptr : Pointer;
 6: begin
 7: Ptr:= SaveIntØØ; GetIntVec( Ø, SaveIntØØ ); SetIntVec( Ø, Ptr );
 8: { usw mit allen umgesetzten Vektoren, siehe Hilfesystem unter
 9: Unit 'System' }
1Ø: end;
11:
12: function GetEnv( Str : String ) : String;
13: type WordPtr = ^ Word; a = array[ Ø..32767 ] of Char;
14: var b : ^a; l, n : Word; TStr : String;
15: label Abbruch1, Abbruch2;
16: begin
17: for l:=1 to Length( Str )
18: do Str[ l ]:= UpCase( Str[ l ] );
19: b:= Ptr( WordPtr( Ptr( PrefixSeg, $2C ) )^, Ø ); TStr:='';
2Ø: for l:=Ø to 32767
21: do begin if b^[l] = Str[1]
22:          then begin for n:=Succ( l ) to Length( Str ) + Pred( l )
23:                     do if b^[ n ] <> Str[ n - Pred( l ) ]
24:                        then Goto Abbruch1;
25:                     for l:= n + 2 to n + 257
26:                     do if b^[l] <> #Ø
27:                        then TStr[l-n-1]:= b^[l]
28:                        else begin TStr[Ø]:= Char( l-n-2 );
29:                                   Goto Abbruch2
30:                             end
31:                end;
32:          Abbruch1:
33:          if ( b^[l] = #Ø ) and ( b^[Succ(l)] = #Ø )
34:          then Goto Abbruch2
35:      end;
36: Abbruch2:
37: GetEnv:= TStr
38: end;
```

Bild 7 Die Prozedur SwapVectors und die Funktion GetEnv für Turbo-Pascal 4.0

gestellt sind, daß beim nächsten Aufruf von Calc.exe innerhalb von *Calculation* die Berechnung fortgesetzt wird. Außerdem wird durch **ey[xi]:=false** vermerkt, daß in dem Punkt mit dem aktuellen Index eine Singularität existiert.

Achtung: Die Exit-Prozedur muß als *far* deklariert sein. Die Steuerung wird dann wieder an die Unit F1 zurückgegeben (Prozedur Calculation) und diese erkennt (xi<>succ(x-max)), daß die Berechnung noch nicht beendet ist, und startet Calc.exe erneut. So ist eine Berechnung von beliebigen, beispielsweise auch gebrochenen rationalen Funktionen ohne die Gefahr eines Programmabbruches durch einen Laufzeitfehler möglich. Damit die Ladezeiten kurz gehalten werden, bietet es sich an, Calc.exe auf die RAM-Disk zu schreiben.

In dem Textfile TPCDir.Txt muß der Pfad für TPC.EXE eingetragen sein. Findet das Programm in diesem Directory TPC.EXE nicht, wird zur Eingabe eines neuen Pfades aufgefordert bis TPC.EXE gefunden wurde (siehe Initialisierungsteil der Unit F1).

Bei Beendigung des Programms werden durch die Exit-Prozedur der Unit F1 automatisch Calc.exe und Calc.pas gelöscht. In Bild 1 wird ein Beispielprogramm gezeigt, in dem die hier angegebenen Routinen zur Tabellierung einer arithmetischen Funktion genutzt werden.

Für viele Probleme ist diese Variante sicher nicht der richtige Weg, da sehr oft nur ein Funktionswert benötigt wird, der dann im Hauptprogramm ausgewertet wird, beispielsweise die Berechnung von Differentialgleichungssystemen. Es wäre unpraktisch, jedesmal ein Programm einladen zu müssen.

Installieren einer Funktion

Wie in dem Programm ProcPtr.pas, © Copyright 1985, 1987 by Borland International, Inc., gezeigt wird, ist es möglich, mehrere Prozeduren variabel aufzurufen:

Procedure CallProc(...);
inline($FF/ $1E/ ProcAddr);
{CALL FAR [ProcAddr]}

Die aufzurufenden Prozeduren müssen eine identische Parameterliste besitzen, ansonsten gerät der Stack durcheinander.

ProcAddr: Pointer muß dazu auf die Adresse der Prozedur zeigen. Da die Prozedur durch *Far-Call* aufgerufen wird, muß sie als *far* deklariert werden, damit am Schluß derselben ein *Far-Ret* steht. Nach /3/ ist auch ein Aufruf von Funktionen wie oben beschrieben möglich:

Function CallFunction(...):...;
inline($FF/ $1E/ FuncAddr);
{CALL FAR [FuncAddr]}

Es ist aber nicht notwendig, daß der Code der Prozedur oder der Funktion innerhalb des Programms steht, von dem sie aufgerufen wird. Daß das mit Interrupt-Prozeduren geht, ist bekannt, mit normalen geht es aber auch. Damit entfällt die Übergabe von Variablen-Adressen, nur die Adresse der Funktion muß diesmal übergeben werden, aber in der anderen Richtung wie in Version 1. Das Programm Calc.exe wird diesmal in *Install* aufgerufen (siehe Bild 4, Zeile 72), und bei der Abarbeitung von Calc.exe wird lediglich die Adresse der Funktion auf den Interrupt-Vektor gelegt, und Calc.exe wird mit **Keep(0)** speicherresident beendet (siehe Bild 5, Zeile 43 bis 44).

Achtung: Keep setzt nicht die Interrupt-Vektoren zurück, deshalb ist auch in *Install* das Tauschen der Interruptvektoren notwendig, damit ein eventueller Laufzeitfehler nicht mit einer Fehlerbehandlung in Calc.exe endet, sondern im Hauptprogramm. Ein Aufruf von Exec (außerhalb von Install) würde nicht den Code der installierten Funktion überschreiben. Um eine neue Funktion zu installieren, muß der alte Code wieder freigegeben werden. Das geschieht mit der Prozedur *ReleaseMemory*. Es ist eine Möglichkeit, um speicherresidente Interruptprogramme sich selbst entfernen zu lassen /3/.

Wenn nicht die Notwendigkeit besteht, die Prozedur *Exec* außerhalb der Prozedur *Install* aufzurufen, muß der installierte Code auch nicht speicherresident gemacht werden, und demzufolge darf dann RealeaseMemory nicht benutzt werden (auch nicht DeallocateProgramm und DeallocateMemory). Es kann also auf den Code eines Programms zugegriffen werden, das bereits beendet wurde und nicht speicherresident ist, denn der Code dieses Programms steht weiterhin im Speicher, bis er mit einem erneuten Aufruf von *Exec* überschrieben wird.

Nach erfolgreicher Ausführung von *Install* (Ergebniswert von Install = true) kann die Funktion y(x) aufgerufen werden, sie liefert nun den Wert f(x) unserer eingegebenen Funktion.

Die installierte Funktion nutzt den Stack des aufrufenden Programms. In Help1.pas muß demzufolge nur Stack für den Ablauf des Programms (Interrupt-Vektor setzen) reserviert werden.

Achtung: Die eventuell mit *Install* deklarierten Variablen ({!Dkt}) der zu installierenden Funktion liegen auf dem Stack. Ihr Inhalt ist beim Aufruf der Funktion unbestimmt. Das Datensegment wird beim Aufruf einer normalen Prozedur oder Funktion von einem anderen Programm aus nicht wie bei einer Interrupt-Prozedur umgesetzt.

Deshalb können keine globalen Variablen deklariert werden; sie können als Referenz Parameter übergeben werden.

In Bild 6 wird ein Beispielprogramm gezeigt, in dem die Routinen zur Tabellierung einer arithmetischen Funktion genutzt werden.

Die Programmteile sollen nur die Technik demonstrieren, sie sind nicht perfekt.

Natürlich ist es auch möglich, mehrere Funktionen oder Funktionen mit anderen Parametern zu installieren:

y(X: Vektor): RealType;

Zur Übergabe von Adressen können bis MS-DOS 3.30 nach /1/ die Interrupt-Vektoren $47-$67 (außer $4A beim PC/AT) benutzt werden.

Deshalb ist das Zurücksetzen der benutzten Interrupt-Vektoren in den Exit-Prozeduren der Units F1 und F2 erforderlich.

Ab Turbo-Pascal 5.0 kann an Stelle des Inline-Codes für den Funktionsaufruf eine prozedurale Variable verwendet werden.

Literatur

/1/ Tischer, M.: PC Intern – Systemprogrammierung. Data Becker GmbH, Düsseldorf 1988, S. 41

/2/ Schäpers, A.: Turbo-Pascal 4.0, Konzepte, Analysen, Tips & Tricks, Vergleich zu 3.0 bis aufs letzte Bit. Addison-Wesley Verlag (Deutschland) GmbH, Bonn 1988

Schulungen

Das Schulungszentrum des Büromaschinenwerkes Sömmerda führt 1990 folgende Softwareschulungen durch:

- Betriebssystem DCP 3.3 (1 Woche)
- Textverarbeitung (1 Woche)
- REDABAS 3 (2 Wochen)
- MULTICALC (1 Woche)
- TPASCAL (2 Wochen)
- POLYCAD (1 Woche)
- SCOM-LAN (1 Woche)

VEB Robotron Büromaschinenwerk „Ernst Thälmann" Sömmerda, Abt. Softwareentwicklung, Weißenseer Straße 52, Sömmerda, 5230; Tel. 4 37 21 oder 4 37 19

Stumpp

Für PC-Neulinge werden Intensivschulungen direkt am PC für die Handhabung der Computer einschließlich ihrer Peripherie, zum Betriebssystem MS-DOS sowie zu Standardsoftware wie Wordstar, dBase, Turbo-Pascal und integrierte Softwarepakete angeboten. Die Schulungszeiten können individuell vereinbart werden.

pd-computerdienste gmbh, Postfach 13, Berlin, 1053; Tel. 4 49 90 95

Dauth

Power on the Chip

Die Familie der 680X0-Prozessoren von Motorola wurde um einen Neuling erweitert, den »68040«. Durch seine 1,2 Millionen Transistoren konnte die Leistung erheblich gesteigert werden.

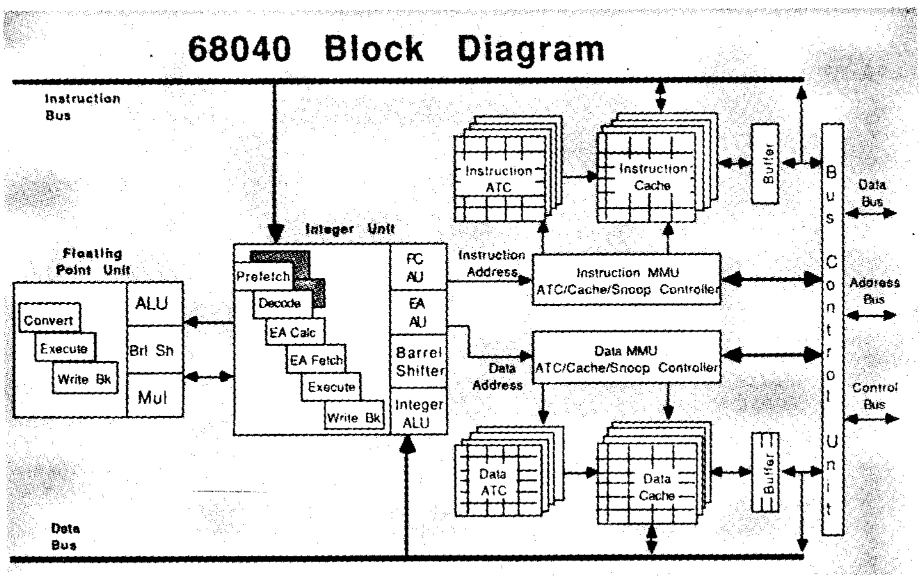
Diesmal fand die Premiere von Motorola völlig antizyklisch in Deutschland statt, dann in weiteren europäischen Ländern und zuletzt kam die USA an die Reihe. Waren dafür nur organisatorische Entscheidungen ausschlaggebend oder kam man diesmal auch der marktpolitischen Bedeutung nach? Von den über 50 Herstellern von Computersystemen, die die Verwendung des 68040 in Betracht gezogen haben, sind allein 34 in Deutschland ansässig. Darunter Firmen wie Apple Computer, Bull, Commodore, Hewlett-Packard/Apollo, Nixdorf, Philips und Unisys.

Mit dem 68040 ist Motorola dem eigenen Gesetz der Software-Kompatibilität treu geblieben. Somit können sämtliche Programme, die auf der 680X0-Familie lauffähig sind, bei einer installierten Basis von ca. 4 Milliarden Dollar, auch mit dem 68040 bearbeitet werden. Dennoch handelt es sich bei dem Chip nicht nur um eine Erweiterung der 680X0-Familie. Der aus 1,2 Millionen Transistor-Funktionen bestehende Prozessor (das sind fast viermal so viel wie beim 68030) wurden nicht nur um weitere Funktionen ergänzt, sondern von Grund auf neu aufgebaut. So bietet der mit 25 MHz getaktete 68040 bereits eine Leistung von 20 MIPS. Nach Herstellerangaben sei er damit bereits um 25 Prozent schneller als beispielsweise der 80486 von Intel. Ähnlich wie bei diesem wurden beim 68040 mehrere Funktionseinheiten zu einem Chip zusammengefaßt. So enthält der 68040 neben einer Integer- auch eine Gleitkomma-Recheneinheit. Mit diesen beiden Einheiten können aufwendige mathematische Berechnungen durchgeführt werden. Die durchschnittliche Fließkomma-Rechenleistung beträgt ca. 3,5 MFLOPS (Million Floating Point Operations per Second). Als Maximum werden 8 MFLOPS vom Hersteller angegeben. Dabei ist diese Einheit mit den Koprozessoren 68881 und 68882 objektcodekompatibel. Besondere Einsatzgebiete sind also nun in Bereichen der Grafikanwendungen, Computersimulationen und Finanzanalysen gegeben. Zur Steigerung der Systemleistung trägt auch der Hardware-Multiplizierer (64 × 64) der Gleitkomma-Recheneinheit bei.

Ergänzt wird das Ganze durch zwei Speicherverwaltungseinheiten (PMMUs) sowie Daten- und Befehlscaches. Von den Speicherverwaltungseinheiten werden sowohl virtuelle Demand-Paging- als auch Echtzeitbetriebssysteme unterstützt. Außerdem ermöglichen sie die gleichzeitige Übersetzung von Befehlen und Operanden.

Um einen hohen Informationsfluß verarbeiten zu können, wurde der 68040 mit zwei, jeweils 4 KByte-großen, Daten- und Befehlscaches ausgestattet. Beide arbeiten gleichzeitig, um Daten und Befehle mit einer Transferrate von bis zu 200 MByte/s dem Prozessor zur Verfügung stellen zu können. Dabei verfolgen sie gleichzeitig Daten auf dem externen Bus, um die Cache-Speicher auf dem neuesten Stand zu halten.

Durch die Zusammenfassung dieser Einheiten auf einem Chip benötigt der Prozessor nur noch ca. 1,3 Taktzyklen für einen Befehl. Eine weitere Besonderheit des Chips: Parallel können 14 Operationen ablaufen. Die Auslieferung des Prozessors in Stückzahlen ist für Ende 1990 vorgesehen. Der Preis für derzeitig erhältliche Muster beträgt ca. 800 Dollar. (ga)



Schematische Darstellung der Funktionseinheiten des 68040

Nachdruck aus Computer Persönlich, Heft 5/1990

Wegbereiter der Informatik

CARL DAVID RUNGE

∗ 1856 Bremen † 1927 Göttingen

Runge hat 1877–80 in Berlin Mathematik und Physik studiert und anschließend promoviert. Schon 3 Jahre später habilitierte er sich und wirkte ab 1883 als Privatdozent in Berlin; 1886 zog er nach Hannover und wurde Professor an der dortigen Technischen Hochschule. Im Jahre 1904 berief man ihn als Ordinarius für Angewandte Mathematik an die Universität Göttingen, wo auch Felix Klein bereits seit 1886 wirkte. Runge bekleidete sein Lehramt bis 1924, er galt als ausgezeichneter Pädagoge. Zwischendurch (1911) weilte er als Austauschprofessor in Amerika.

Runge leistete bedeutende Arbeiten zur numerischen Anwendung mathematischer Verfahren auf technische Probleme; er erfand eine große Anzahl zweckmäßiger Methoden, um mit kleinstem Aufwand an numerischer Arbeit zur gewünschten approximativen Lösung komplizierter Probleme gelangen zu können. Er hat damit zur computergestützten Auswertbarkeit mathematischer Aufgabenstellungen wesentlich mit beigetragen. Wohl jeder mathematisch ausgebildete Programmierer kennt zum Beispiel die Formeln von Runge-Kutta zur approximativen Berechnung eines partikulären Integrals für Differentialgleichungen der Form

$$y' = f(x,y)$$

wobei nur Werte von f(x,y) benutzt werden (1895). Dieses Runge-Kutta-Verfahren, ein numerisch stabiles extrapolatives Einschrittverfahren ist eines der wichtigsten numerischen Lösungsverfahren für gewöhnliche Differentialgleichungen geworden und dürfte heute zur Standardausrüstung jeder anspruchsvollen Computer-Software gehören.

Ein weiteres wesentliches Arbeitsgebiet Runges war die Ausgleichsrechnung (1897), die das Ziel verfolgt, etwa mit Hilfe der Methode der kleinsten Quadrate aus fehlerbehafteten Meßwerten Näherungswerte für die zu messenden Größen zu gewinnen und deren Genauigkeit anzugeben. Außerdem befaßte sich Runge mit der Approximation und Interpolation von Funktionen (Satz von Runge, 1901), mit harmonischer Analyse (1902) sowie mit der Summation trigonometrischer Polynome. Er hat auch praktikable Methoden für graphisches Rechnen entwickelt (graphische Lösung algebraischer Gleichungen und partieller Differentialgleichungen) und darüber – neben anderem – ein Lehrbuch verfaßt.

In seinen späteren Lebensjahren wandte sich Runge stärker der Physik zu (Messung von Spektrallinien in der Optik) und widmete sich auch ganz speziellen praktischen Problemen: So entwickelte er Ortsbestimmungsverfahren auf See und im Ballon und löste Probleme aus der Festigkeitslehre.

Dr. Klaus Biener

Computer-Club

Typisierte CP/M-Dateien und Turbo-Pascal 5.0

Eberhard Schmidt, Frankfurt (Oder)

Da in Turbo-Pascal beim Übergang von CP/M zu MS-DOS der Aufbau der typisierten Dateien stark geändert wurde, stehen Programmierer, die häufig zwischen beiden Betriebssystemen wechseln müssen, vor einigen Problemen.

Unter CP/M ist nur die Anzahl der 128-Byte-Records einer Datei bekannt. Deshalb setzt Turbo-Pascal hier seinen typisierten Dateien vier Byte voran, in denen die Anzahl der Datensätze und die Satzgröße vermerkt ist. Diese Kenngrößen entfallen bei Dateien, die von den MS-DOS-Versionen von Turbo-Pascal erzeugt werden, weil die Satzanzahl sich durch die Kenntnis der exakten Dateigröße unter MS-DOS aus der Satzgröße errechnen läßt.
Es ist zwar für den Nutzer kein Problem, beim einmaligen Übergang von CP/M zu MS-DOS seine typisierten Dateien durch ein kleines Konvertierungsprogramm von diesen vier Byte zu befreien, belastend wird die Angelegenheit jedoch bei einem ständigen Wechsel zwischen beiden Betriebssystemen. Das Laden entsprechender Treiber ermöglicht den Zugriff auf CP/M-Dateien auch unter MS-DOS. Im Interesse eines Programmnutzers soll sein Pascal-Programm nun auch die auf dieser Diskette befindlichen typisierten Pascal-Dateien bearbeiten können. Damit soll er davon entlastet werden, parallele Datenbestände zu führen. Das Nutzerprogramm muß dazu die Aufgaben übernehmen, die in der CP/M-Version von Turbo-Pascal die Laufzeitbibliothek erledigt hat: die Überlagerung einer physischen CP/M-Datei mit der logischen Satzstruktur des Nutzerprogramms. Die Prozeduren und Funktionen der Unit *CPM* (Bild 1) ermöglichen dies, indem sie die Datei beim Zugriff als nicht typisiert ansehen und den Datenversatz von vier Byte berücksichtigen. Um in Programmen mit dieser Unit wahlweise CP/M- oder MS-DOS-Dateien bearbeiten zu können, sind im Interface-Teil prozedurale Variablen definiert, denen beim Programmstart die Adressen der Module für den Zugriff auf CP/M-Dateien zugewiesen werden. Diese Variablen werden wie die Prozeduren der Unit System benutzt. Ausnahmen bilden nur die Prozedurvariable *_Rewrite* und der Typ der Dateivariablen. Die Variable *_Rewrite* verlangt als zweiten Parameter die Byteanzahl eines Datensatzes, die Dateivariable muß dem in der Unit deklarierten Datentyp *TypeFile* entsprechen.
Zusätzlich wird die Funktionsvariabl-

```
  1:  UNIT CPM { Version 9/89 } ; INTERFACE
  2:  { E.Schmidt, Beerenweg 14, Frankfurt/Oder, 1200 }
  3:  PROCEDURE cpm_Prozedur_Zuweisung ;
  4:  TYPE _TypeFile = RECORD
  5:                     Block                 : FILE ;
  6:                     Size , Count , Posi   : WORD   ;
  7:                   END ;
  8:        _AssignTyp   = PROCEDURE ( VAR f ; s : STRING    ) ;
  9:        _RewriteTyp  = PROCEDURE ( VAR f ; s : WORD      ) ;
 10:        _SeekTyp     = PROCEDURE ( VAR f ; s : LONGINT   ) ;
 11:        _ResetTyp    = PROCEDURE ( VAR f                 ) ;
 12:        _ReadTyp     = PROCEDURE ( VAR f , s             ) ;
 13:        _FileSizeTyp = FUNCTION  ( VAR f ) : LONGINT ;
 14:        _FilePosTyp  = FUNCTION  ( VAR f ) : LONGINT ;
 15:        _EofTyp      = FUNCTION  ( VAR f ) : BOOLEAN ;
 16:  VAR   _Assign                            : _AssignTyp   ;
 17:        _Rewrite                           : _RewriteTyp  ;
 18:        _Reset    , _Close    , _Truncate  : _ResetTyp    ;
 19:        _Seek                              : _SeekTyp     ;
 20:        _Read     , _Write                 : _ReadTyp     ;
 21:        _FileSize , _FilePos               : _FileSizeTyp ;
 22:        _Eof      , _FileExist             : _EofTyp      ;
 23:  IMPLEMENTATION {$F+}
 24:  VAR Puffer : ARRAY [ 0 .. 127 ] OF BYTE ;
 25:      buffer : ARRAY [ 0 ..  63 ] OF INTEGER
 26:               ABSOLUTE Puffer ;
 27:  FUNCTION BytePosition ( filvar : _TypeFile ) : LONGINT ;
 28:  BEGIN WITH filvar DO
 29:     BytePosition := LONGINT ( Posi ) * Size + 4
 30:  END ;
 31:
 32:  { ---------------------- Assign --------------------- }
 33:  PROCEDURE cpm_Assign ( VAR f ; n : STRING ) ;
 34:  VAR filvar : _TypeFile ABSOLUTE f ;
 35:  BEGIN ASSIGN ( filvar . Block , n ) END ;
 36:  { ---------------------- Rewrite -------------------- }
 37:  PROCEDURE cpm_Rewrite ( VAR f ; sl : WORD ) ;
 38:  VAR filvar : _TypeFile ABSOLUTE f ;
 39:  BEGIN WITH filvar DO BEGIN
 40:     REWRITE ( Block ) ;
 41:     Count := 0 ;  Posi := 0 ;  Size  := sl ;
 42:     buffer [ 0 ] := Count ; buffer [ 1 ] := Size ;
 43:     BLOCKWRITE ( Block , buffer , 1 )
 44:  END END ;
 45:  { ----------------------- Reset --------------------- }
 46:  PROCEDURE cpm_Reset ( VAR f ) ;
 47:  VAR filvar : _TypeFile ABSOLUTE f ; k : LONGINT ;
 48:  BEGIN WITH filvar DO BEGIN
 49:     RESET ( Block ) ;
 50:     IF FILESIZE ( Block ) = 0 THEN RUNERROR ( $90 ) ;
 51:     BLOCKREAD ( Block , buffer , 1 ) ;
 52:     Size := buffer [ 1 ] ; Posi := 0 ;
 53:     Count := buffer [ 0 ] ; k := Count * Size + 3 ;
 54:     IF k DIV 128 >= FILESIZE ( Block )
 55:        THEN RUNERROR ( $99 )
 56:  END END ;
 57:  { ---------------------- Seek ----------------------- }
 58:  PROCEDURE cpm_Seek ( VAR f ; Pos : LONGINT ) ;
 59:  VAR filvar : _TypeFile ABSOLUTE f ;
 60:  BEGIN WITH filvar DO BEGIN
 61:     IF Pos > Count THEN RUNERROR ( $91 ) ; Posi := Pos ;
 62:     SEEK ( Block , BytePosition ( filvar ) DIV 128 )
 63:  END END ;
 64:  { ---------------------- Read ----------------------- }
 65:  PROCEDURE cpm_Read ( VAR f , Satz ) ;
 66:  VAR filvar : _TypeFile ABSOLUTE f ;
 67:  PROCEDURE Satz_Lesen ;
 68:  VAR sa : ARRAY [ 0 .. 0 ] OF BYTE ABSOLUTE Satz ;
 69:     i , m : INTEGER ;   j : BYTE ;  k : LONGINT ;
 70:  BEGIN WITH filvar DO BEGIN
 71:     k := BytePosition ( filvar ) ;    j := k MOD 128 ;
 72:     SEEK ( Block , k DIV 128 ) ;
 73:     BLOCKREAD ( Block , Puffer , 1 ) ;
 74:     IF j + Size > 128 THEN m := 128 - j ELSE m := Size ;
 75:     MOVE ( Puffer [ j ] , sa , m ) ; i := Size - m ;
 76:     WHILE i >= 128 DO
 77:        BEGIN
 78:         BLOCKREAD ( Block , Puffer , 1 ) ;
 79:         MOVE ( Puffer , sa [ m ] , 128 ) ;
 80:         INC ( m , 128 ) ; DEC ( i , 128 )
 81:        END ;
 82:     IF i > 0 THEN
 83:        BEGIN
 84:         BLOCKREAD ( Block , Puffer , 1 ) ;
 85:         MOVE ( Puffer , sa [ m ] , i )
 86:        END ;
 87:     INC ( Posi )
 88:  END END ;
 89:  BEGIN WITH filvar DO BEGIN
 90:     IF Posi < Count THEN Satz_Lesen ELSE RUNERROR ( $91 )
 91:  END END ; { Read }
 92:  { ---------------------- Truncate ------------------- }
 93:  PROCEDURE cpm_Truncate ( VAR f ) ;
 94:  VAR filvar : _TypeFile ABSOLUTE f ;
 95:     i         : LONGINT                ;   j : WORD ;
 96:  BEGIN WITH filvar DO BEGIN
 97:     i := BytePosition ( filvar ) ; j := i DIV 128 ;
 98:     IF i MOD 128 > 0 THEN INC ( j ) ; Count := Posi ;
 99:     SEEK ( Block , j ) ; TRUNCATE ( Block ) ;
100:  END END ; { Truncate }
101:  { ---------------------- Close ---------------------- }
102:  PROCEDURE cpm_Close ( VAR f ) ;
103:  VAR filvar : _TypeFile ABSOLUTE f ;
104:  BEGIN WITH filvar DO BEGIN
105:     SEEK ( Block , 0 ) ;
106:     BLOCKREAD  ( Block , buffer , 1 ) ;
107:     buffer [ 0 ] := Count ;  SEEK ( Block , 0 ) ;
108:     BLOCKWRITE ( Block , buffer , 1 ) ;  CLOSE ( Block )
109:  END END ;
110:  { ---------------------- FileSize ------------------- }
111:  FUNCTION cpm_FileSize ( VAR f ) : LONGINT ;
112:  VAR filvar : _TypeFile ABSOLUTE f ;
113:  BEGIN cpm_FileSize := filvar . Count END ;
114:  { ---------------------- FilePos -------------------- }
115:  FUNCTION cpm_FilePos ( VAR f ) : LONGINT ;
116:  VAR filvar : _TypeFile ABSOLUTE f ;
117:  BEGIN cpm_FilePos  := filvar . Posi  END ;
118:  { ---------------------- Eof ------------------------ }
119:  FUNCTION cpm_Eof ( VAR f ) : BOOLEAN ;
120:  VAR filvar : _TypeFile ABSOLUTE f ;
121:  BEGIN WITH filvar DO cpm_Eof := Posi = Count END ;
122:  { ---------------------- Write ---------------------- }
123:  PROCEDURE cpm_Write ( VAR f , Satz ) ;
124:  VAR filvar : _TypeFile ABSOLUTE f ;
125:  PROCEDURE Satz_Schreiben ;
126:  VAR sa : ARRAY [ 0 .. 0 ] OF BYTE ABSOLUTE Satz ;
127:     i , m : INTEGER ;   j : BYTE ;  k : LONGINT ;
128:  BEGIN WITH filvar DO BEGIN
129:    k := BytePosition ( filvar ) ;
130:    i := k DIV 128 ;  j := k MOD 128 ;
131:    SEEK ( Block , i ) ;
132:    IF ( j > 0 ) OR ( SUCC ( Posi ) < Count ) THEN
133:     BEGIN
134:      BLOCKREAD ( Block , Puffer , 1 ) ; SEEK ( Block , i )
135:     END ;
136:    IF j + Size > 128 THEN m := 128 - j ELSE m := Size ;
137:    MOVE ( sa , Puffer [ j ] , m ) ;
138:    BLOCKWRITE ( Block , Puffer , 1 ) ; i := Size - m ;
139:    WHILE i >= 128 DO
140:       BEGIN
141:        MOVE ( sa [ m ] , Puffer , 128 ) ;
142:        BLOCKWRITE ( Block , Puffer , 1 ) ;
143:        INC ( m , 128 ) ; DEC ( i , 128 )
144:       END ;
145:    IF i > 0 THEN
146:       BEGIN
147:        IF SUCC ( Posi ) < Count THEN
148:           BEGIN
149:            BLOCKREAD ( Block , Puffer , 1 ) ;
150:            SEEK ( Block , PRED ( FILEPOS ( Block ) ) )
151:           END ;
152:        MOVE ( sa [ m ] , Puffer , i ) ;
153:        BLOCKWRITE ( Block , Puffer , 1 )
154:       END ;
155:    IF Posi = Count THEN INC ( Count ) ; INC ( Posi )
156:  END END ;
157:  BEGIN
158:     IF filvar . Posi = $FFFF
159:        THEN RUNERROR ( $F2 ) ELSE Satz_Schreiben
160:  END ; { Write }
161:  { ---------------------- FileExist ------------------ }
162:  FUNCTION cpm_FileExist ( VAR f ) : BOOLEAN ;
163:  VAR filvar : _TypeFile ABSOLUTE f ;
164:  BEGIN WITH filvar DO BEGIN
165:     {$I-} RESET ( Block ) ; {$I+} cpm_FileExist := TRUE ;
166:     IF IORESULT = 0
167:        THEN CLOSE ( Block ) ELSE cpm_FileExist := FALSE
168:  END END ;
169:  { ---------------- INTERFACE-Prozedur --------------- }
170:  PROCEDURE cpm_Prozedur_Zuweisung ;
171:  BEGIN
172:     _Assign := cpm_Assign  ; _Rewrite   := cpm_Rewrite  ;
173:     _Reset  := cpm_Reset   ; _Write     := cpm_Write    ;
174:     _Seek   := cpm_Seek    ; _FileSize  := cpm_FileSize ;
175:     _Eof    := cpm_Eof     ; _FilePos   := cpm_FilePos  ;
176:     _Read   := cpm_Read    ; _Truncate  := cpm_Truncate ;
177:     _Close  := cpm_Close   ; _FileExist := cpm_FileExist;
178:  END ;
179:  { ----- Initialisierung der INTERFACE-Variablen ----- }
180:  BEGIN cpm_Prozedur_Zuweisung END .
```

***Bild 1** Unit zur Arbeit mit typisierten CP/M-Dateien*

_FileExist bereitgestellt, die den Wert *True* liefert, wenn die angesprochene Datei vorgefunden wurde. Die Unit arbeitet mit automatischer Fehlerkontrolle. Die für typisierte Dateien relevanten Laufzeitfehler führen zum Programmabbruch mit den üblichen Fehlernummern. Bild 3 zeigt ein Programm zur oben erwähnten einmaligen Übernahme einer CP/M-Datei auf MS-DOS-Niveau. Der Programmausschnitt in Bild 4 soll den über die Variable *CPM_Datei* gesteuerten alternativen Dateizugriff demonstrieren.

```
 1:  {$F+}    (* Datei DOSPROC.INC *)
 2:  PROCEDURE dos_Assign ( VAR f ; s : STRING ) ;
 3:  VAR filvar : FILE OF filtyp ABSOLUTE f ;
 4:  BEGIN ASSIGN ( filvar , s ) END ;
 5:  PROCEDURE dos_Rewrite ( VAR f : dummy : WORD ) ;
 6:  VAR filvar : FILE OF filtyp ABSOLUTE f ;
 7:  BEGIN REWRITE ( filvar ) END ;
 8:  PROCEDURE dos_Reset ( VAR f ) ;
 9:  VAR filvar : FILE OF filtyp ABSOLUTE f ;
10:  BEGIN RESET ( filvar ) END ;
11:  PROCEDURE dos_Read ( VAR f , s ) ;
12:  VAR filvar : FILE OF filtyp ABSOLUTE f ;
13:      satz   : filtyp ABSOLUTE s ;
14:  BEGIN READ ( filvar , satz ) END ;
15:  PROCEDURE dos_Truncate ( VAR f ) ;
16:  VAR filvar : FILE OF filtyp ABSOLUTE f ;
17:  BEGIN TRUNCATE ( filvar ) END ;
18:  PROCEDURE dos_Write ( VAR f , s ) ;
19:  VAR filvar : FILE OF filtyp ABSOLUTE f ;
20:      satz   : filtyp ABSOLUTE s ;
21:  BEGIN WRITE ( filvar , satz ) END ;
22:  PROCEDURE dos_Close ( VAR f ) ;
23:  VAR filvar : FILE OF filtyp ABSOLUTE f ;
24:  BEGIN CLOSE ( filvar ) END ;
25:  PROCEDURE dos_Seek ( VAR f ; Pos : LONGINT ) ;
26:  VAR filvar : FILE OF filtyp ABSOLUTE f ;
27:  BEGIN SEEK ( filvar , Pos ) END ;
28:  FUNCTION dos_FileSize ( VAR f ) : LONGINT ;
29:  VAR filvar : FILE OF filtyp ABSOLUTE f ;
30:  BEGIN dos_FileSize := FILESIZE ( filvar ) END ;
31:  FUNCTION dos_FilePos ( VAR f ) : LONGINT ;
32:  VAR filvar : FILE OF filtyp ABSOLUTE f ;
33:  BEGIN dos_FilePos := FILESIZE ( filvar ) END ;
34:  FUNCTION dos_Eof ( VAR f ) : BOOLEAN ;
35:  VAR filvar : FILE OF filtyp ABSOLUTE f ;
36:  BEGIN dos_Eof := EOF ( filvar ) END ;
37:  FUNCTION dos_FileExist ( VAR f ) : BOOLEAN ;
38:  VAR filvar : FILE OF filtyp ABSOLUTE f ;
39:  BEGIN
40:    {$I-} RESET ( filvar ) ; {$I+}
41:    IF IORESULT = 0
42:      THEN
43:       BEGIN
44:          CLOSE ( filvar ) ; dos_FileExist := TRUE
45:       END
46:      ELSE dos_FileExist := FALSE
47:  END ;
48:  {$F-}
49:  PROCEDURE dos_Prozedur_Zuweisung ;
50:  BEGIN
51:     _Assign   := dos_Assign ; _Rewrite   := dos_Rewrite   ;
52:     _Reset    := dos_Reset  ; _Write     := dos_Write     ;
53:     _Seek     := dos_Seek   ; _FileSize  := dos_FileSize  ;
54:     _Eof      := dos_Eof    ; _FilePos   := dos_FilePos   ;
55:     _Read     := dos_Read   ; _Truncate  := dos_Truncate  ;
56:     _Close    := dos_Close  ; _FileExist := dos_FileExist
57:  END ;
```

▲ Bild 2
Include zur Arbeit mit typisierten MS-DOS-Dateien

```
 1:  USES CPM ;
 2:  TYPE filtyp = RECORD
 3:                  lfd_Nr   : WORD ;
 4:                  Bezeich : STRING [ 50 ] ;
 5:                  Preis    : REAL
 6:                END ;
 7:  VAR  Satz   : filtyp        ;
 8:       source : _TypeFile     ;
 9:       destin : FILE OF filtyp ;
10:       i      : INTEGER       ;
11:  BEGIN
12:    _Assign ( source ,  QUELLE' ) ;
13:    IF _FileExist ( source ) THEN
14:     BEGIN
15:      _Reset  ( source ) ;
16:      ASSIGN  ( destin , 'ZIEL' ) ;
17:      REWRITE ( destin ) ;
18:      FOR i := 1 TO _FileSize ( source ) DO
19:       BEGIN
20:         _Read ( source , Satz ) ;
21:         WRITE ( destin , Satz )
22:       END ;
23:      _Close ( source ) ;
24:      CLOSE ( destin )
25:     END
26:  END.
```

Bild 3
Beispiel zur Konvertierung einer CP/M- in eine MS-DOS-Datei

```
 1:  USES CPM ;
 2:  TYPE filtyp = RECORD
 3:                  lfd_Nr   : WORD ;
 4:                  Bezeich : STRING [ 50 ] ;
 5:                  Preis    : REAL
 6:                END ;
 7:  {$I DOSPROC.INC}
 8:  VAR  Satz        : filtyp      ;
 9:       filvar      : _TypeFile   ;
10:       i           : INTEGER     ;
11:       CPM_Datei : BOOLEAN       ;
12:  BEGIN
13:    CPM_Datei := TRUE ;
14:    IF CPM_Datei
15:       THEN cpm_Prozedur_Zuweisung
16:       ELSE dos_Prozedur_Zuweisung ;
17:    _Assign  ( filvar , 'TESTDATE' ) ;
18:    _Rewrite ( filvar , SIZEOF ( Satz ) ) ;
19:    Satz . Bezeich [ 0 ] := #50 ;
20:    FOR i := 1 TO 50 DO
21:       WITH Satz DO BEGIN
22:        lfd_Nr := i ;
23:        FILLCHAR ( Bezeich [ 1 ] , 50 , i + 32 )
24:        Preis := 100 * i ;
25:        _Write ( filvar , Satz )
26:       END ;
27:    _Seek ( filvar , 0 ) ;
28:    WHILE NOT _Eof ( filvar ) DO WITH Satz DO
29:       BEGIN
30:        _Read ( filvar , Satz ) ;
31:        WRITELN ( lfd_Nr:6 , Bezeich:52 ,
32:                  Preis:8:2 )
33:       END ;
34:    _Close ( filvar )
35:  END .
```

Bild 4 Beispiel zum wahlweisen Zugriff auf CP/M- oder MS-DOS-Dateien

Hinweis für KC 85-Nutzer

Die Überarbeitung des Programms CO134 Buchhaltung, das durch den VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen auf Kassette vertrieben wird, hatte die Änderung der Endadressen zur Folge, so daß die aktuelle Programmversion und die Beschreibung nicht übereinstimmen. Wir erhielten dazu vom Hersteller die folgende Information:

Die auf Seite 5 der Programmbeschreibung angegebenen Endadressen der Programme Buchh/3D und Buchh/4D zum Abspeichern auf Diskette entsprechen nicht dem aktuellen Stand der vorliegenden Programmversion.

Wir empfehlen daher dem Anwender, die auf dem Bildschirm angezeigten Adressen beim Einladen der Programme von der Kassette als Parameter zum Abspeichern auf die Diskette zu verwenden, da sonst ein fehlerfreier Ablauf der Programme bei einem Wiederstart von der Diskette nicht gewährleistet werden kann.

✉ KONTAKT ☎

VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen, Eisenacher Str. 40, Mühlhausen, 5700; Tel. 5 32 88

Turbo-Pascal-Tip

Hexadezimalverarbeitung

In einigen Anwendungsfällen ist es sinnvoll, in der Ein- und Ausgabe mit Hexadezimalzahlen zu arbeiten. Dafür bieten höhere Programmiersprachen jedoch selten die entsprechenden Routinen an. Im folgenden möchte ich Ihnen eine Lösung für dieses Problem in Turbo-Pascal anbieten.

Die Prozedur *Readhex* liest einen String ein und prüft, ob dieser String der hexadezimalen Schreibweise entspricht und gibt den erhaltenen Wert als Integer zurück. Bei falscher Eingabe wird die Aufforderung wiederholt. Wird eine Leereingabe registriert, gibt diese Prozedur den Wert 0 zurück. Wenn auch Null als Wert benötigt wird, besteht die Möglichkeit, die Boolesche Variable *Gueltig* in die Parameterliste mit aufzunehmen, um das Ergebnis außerhalb dieser Prozedur auswerten zu können. Die Funktion *POT* kann in vielen anderen Programmen Verwendung finden, da sie nicht als Standardfunktion in Turbo-Pascal enthalten ist. Es gilt: $POT = e1^{e2}$.

Die Ausgabe von Integerzahlen in hexadezimaler Schreibweise wird durch die Funktion *HEX* ermöglicht. Hier wurde jedoch nicht die eleganteste, sondern die schnellste Art der Umwandlung gewählt. *Hartmut Voigt*

```
{READHEX  liest  hexadezimale  Ziffern  ein und  gibt  sie  als  integer
          zurueck }

procedure readhex(var aus:integer);
var ep:string[4];m,i,j,k:integer;gueltig:boolean;

function pot(e1,e2:real):real;
begin pot:=exp(e2*ln(e1));end;

begin
  repeat
    gueltig:=true;
    aus:=0;
    buflen:=4;
    read(ep);
    k:=0;
    for i:=1 to length(ep) do ep[i]:=upcase(ep[i]);
    if ep<>'' then
    begin
     repeat
       k:=k+1;
       if gueltig then gueltig:=ep[k] in ['0'..'9','A'..'F'];
     until k=length(ep);                 {test auf hex-schreibweise}
      if gueltig then
      begin
        i:=0;
        m:=length(ep);
        for i:=1 to length(ep)  do
        begin
          m:=m-1;
          if ep[i] in['0'..'9'] then val(ep[i],j,k)
                                else
                                begin
                                  j:=ord(ep[i]);
                                  j:=j-$37
                                end;
          aus:=aus+(j*round(pot(16,m)));
        end;
      end else for i:=1 to length(ep) do write(^H);
    end;
  until gueltig or (ep='');
  if ep='' then aus:=0;
end;

{HEX ergibt den integerwert <a> als vierstelliger string   }

function hex(a:integer):str4;
const chiffre:array[0..15]of char=('0','1','2','3','4','5','6',
                                   '7','8','9','A','B','C','D','E','F');
var h:byte; kx:str4;
begin
   h:=hi(a) and $f0; h:=h shr 4; kx:=chiffre[h];
   h:=hi(a) and $0f; kx:=kx+chiffre[h];
   h:=lo(a) and $f0; h:=h shr 4; kx:=kx+chiffre[h];
   h:=lo(a) and $0f; kx:=kx+chiffre[h];
   hex:=kx
end;
```

MP

Börse

Festplattenüberwachung

Das Betriebssystem DCP unterstützt das Erkennen von unbeabsichtigten Veränderungen/Verfälschungen eines Datenträgers nur ungenügend. Die Programme **DHEDL** und **DHPDL** wurden mit dem Ziel entwickelt, solche Veränderungen/Verfälschungen vor dem Auftreten schwerwiegender Fehler (z. B. vor der Sicherung der Dateien) zu erkennen.
DHEDL durchsucht einen gesamten Datenträger (auch Hidden- und SYS-Dateien) nach zu analysierenden Dateien und erzeugt eine Datenliste, die die wichtigsten Kennwerte der Datei enthält. **DHPDL** verwendet die durch DHEDL erzeugte Datenliste und vergleicht deren Angaben mit den tatsächlichen Werten auf dem Datenträger. Dabei werden folgende Abweichungen im Prüfprotokoll festgehalten: Veränderung der Dateigröße, der Dateiattribute, der Datei-Zeitangabe, Fehlen der Datei und neu hinzugekommene Dateien.

VEB Datenverarbeitungszentrum Halle, Abt. FP, Block 081, Halle-Neustadt, 4090 *Dierich*

Turbo-Konvertierung von CP/M nach MS-DOS

In unserem Betrieb ergab sich die Forderung, an einem A 7100 mit Turbo-Pascal erzeugte Dateien an MS-DOS-fähiger Gerätetechnik weiter verarbeiten zu können und die Resultate später erneut unter CP/M-Steuerung zur Verfügung zu haben. Die entscheidende Schwierigkeit dabei war die Überbrückung bzw. Erzeugung der Vier-Byte-Offsets, wie sie am Anfang der unter CP/M erzeugten Turbo-Dateien auftreten. Da beim Konvertieren in beiden Richtungen bis auf eine Ausnahme Quelle und Ziel als typenlose Dateien behandelt wurden, ergab sich zwangsläufig auch ein in der Abarbeitung sehr effizientes Programm.
Die Nachnutzung ist auf der Basis einer Kurzdokumentation und des Programmausdruckes problemlos möglich.

VEB Erfurter Malzwerke, Iderhoffstraße 3, Erfurt, 5010; Tel. 5 15 61, App. 66 *Heydecke*

Verteilter Compiler comFORTH-plus

Der verteilte Compiler comFORTH-plus ist ein leistungsfähiges Werkzeug zur interaktiven Programmierung verteilter und eingebetteter Mikrorechnersysteme. Voraussetzung für seinen Einsatz ist die Verfügbarkeit eines 8- oder eines 16-Bit-Rechners mit einem CP/M- oder MS-DOS-kompatiblen Betriebssystem als Entwicklungsrechner. Unterstützt werden die Prozessorfamilien Z 8, Z 80, 8086 sowie TMS320. comFORTH-plus stellt folgende Leistungen zur Verfügung: interaktives Programmieren und Testen eingebetteter Mikrorechner mit dem Komfort eines Personalcomputers, simultanes Programmieren und Testen mehrerer Zielsysteme, Einbeziehung der Funktionseinheiten mehrerer Rechnersysteme in ein Programm zur Koordinierung des Gesamtsystems.

FORTecH Software, Albert-Einstein-Straße 2, Rostock 6, 2500; Tel. 40 52 02 *Dr. Pfüller*

Kopplung A 7150 – DZT 120 × 90/RS

Wir bieten ein Programm zur Ausgabe von CADdy-Plotfiles auf dem Digitalzeichentisch DZT 120 × 90/RS (Carl Zeiss JENA) über die COM2- oder COM3-Schnittstelle zur Nachnutzung an. Getestet wurde das Programm auf einem A 7150 unter PC-DOS 3.3 mit CADdy V 3.04.
Wir suchen Technologen-Software (Maschinenbau/Gerätebau) zur Normzeit- und MVN-Berechnung für Schleifen, Fräsen, konventionelles Drehen, Gewindeschneiden (außen) u. a. für A 7150 oder K 1630.

VEB Baumwollspinnerei Leinefelde, Abt. RMB, Birkunger Straße, Leinefelde, 5600; Tel. 81 38 35 *Kozlowski*

Synchroner Adapter

Mit einem bei uns entwickelten Adapter besteht die Möglichkeit, 16-Bit-Personalcomputer der XT/AT-Klasse (EC 1834 mit direktem Steckverbinder) an die zentrale ESER-II-Rechentechnik anzuschließen. Mit einem speziellen Programm kann Terminalemulation (3270) sowie unter Nutzung des SVS 7.2 M 1 Dateitransfer (3780) betrieben werden. Dieser Adapter für die binäre synchrone Kommunikation (BSC) ist als Steckeinheit ausgelegt und in einem freien Slot des PC-Busses zu nutzen. Erfolgt der Anschluß des Personalcomputers über TSR/A (EC 8404 M), so besteht die Notwendigkeit, daß der vom ESER ausgesandte DKOI-Code zur Verarbeitung bis zum Endgerät gelangen kann.

Möglicher Lieferumfang: unbestückter BSC-Adapter mit Dokumentation, Stücklisten u. a., modifiziertes erweitertes TSR/A-Steuerprogramm

VEB Stahl- und Walzwerk Brandenburg, Abt. DV 3, Koll. Detlev Voigt, Straße der Aktivisten, Brandenburg, 1800; Tel. 55 26 15 *Schober*

Programmiertool PULLDOWN

Wir bieten mit dem Kommunikationsmodul PULLDOWN ein Programmtool an, das Pulldown-Menüs und Formulare verwaltet und somit eine universelle Nutzerschnittstelle für viele Bereiche der Softwareentwicklung darstellt. Folgende Funktionen werden durch den Modul ermöglicht: Generierung und Verwaltung von Einzelmenüs und Menübäumen sowie von Formularen, Zugriffsmöglichkeit des Anwendungsprogrammierers auf grundlegende Routinen des PULLDOWN-Systems.
Der Modul wurde in Turbo-Pascal 5.0 geschrieben und wird als Unit für den Anwender nutzbar sein. Im Programmpaket sind enthalten: die Unit PULLDOWN, ein Editor für die Formulardaten, ein Installationsprogramm für die Menüfarben und eine Beschreibung der Units. Anpassungen des Kommunikationsmoduls an spezielle Kundenwünsche werden nach Absprache ausgeführt.

VEB Starkstrom-Anlagenbau Leipzig, Abt. TM 2, Schützenstraße 4–6, Leipzig, 7010; Tel. 7 96 10 *Müller*

AutoCAD-Zeichnungen auf DZT 1200/900 RS

Das in einer 8- und einer 16-Bit-Version vorliegende Programm **ACADZT** ermöglicht das Zeichnen von AutoCAD-Plotdateien auf DZT, K 6418 u. ä. Geräten außerhalb von AutoCAD. Es können ferner plotterspezifische Parameter (Geschwindigkeit, Auswahl Zeichenstift, Lage Nullpunkt) definiert werden.

VEB Starkstrom-Anlagenbau Magdeburg, Abt. TEC, Blankenburger Straße 58/70, Magdeburg, 3014; Tel 4 84 58 *Darge*

IPC-Forth-System für 8-Bit-P8000

Die IPC-Forth-Systemsoftware Version 1.2 dient zur Nutzung eines 8-Bit-P8000 als Steuerrechner für die Bilderkennungseinheit BEE 1010. Die Lösung gestattet die Nutzung von Anwendersoftware für die BEE 1010 auf dem 8-Bit-P8000, die mit BS/PC entwickelt wurde. Voraussetzung: IPC-Forth-Systemsoftware, BEE 1010 (VEB Studiotechnik Berlin) und P8000 OS/M Version 3.0.

VEB Fernsehkolbenwerk Friedrichshain/Tschernitz, Abt. Technologische Entwicklung, Koll. Schulz, Tschernitz, 7591; Tel. 43 04 *Dr. Hemmann*

Parameterübergabe von Turbo-Pascal 4.0 an dBase III Plus

Die Parameterübergabe von Turbo-Pascal 4.0 an dBase III Plus ist nur über Dateien möglich, da dBase III Plus nicht mehr über die Funktionen PEEK und POKE verfügt und auch sonst kein Zugriff auf interne DOS-Funktionen vorgesehen ist. Dieses Problem ließ sich über das Environment des dBase-Programms lösen. In dBase III Plus gibt es eine Funktion GETENV (parameter), welche den Wert des Umgebungsparameters (parameter) aus dBase III Plus liefert. Es wurde eine Möglichkeit gefunden, den Parameter in das Environment des dBase III Plus-Programms einzutragen. Man kann den übergebenen Parameter über die Funktion GETENV in das dBase-Programm übernehmen. Das erfolgt mit der Turbo-Pascal-Unit DBPARA.TPU.

Staatliches Versorgungskontor für Pharmazie und Medizintechnik Berlin, ORZ, Projektierungsgruppe Schwerin, Baustraße 3, Schwerin, 2771; Tel. 55 81, App. 50 *Boyde*

Modulbibliothek und Programmierhilfen für Turbo-Pascal

Zu der im Heft 8/89 veröffentlichten Bibliothek bieten wir folgende Module an:
AUSWAHL-BOX Auswahl über Nummernmenü analog POWER
DB-BOX/DBR-BOX schnelles sequentielles Ausgeben/Lesen von REDABAS-Dateien (ab II aufwärts)
DCP-BOX systemspezifische Kommandos wie für Cursormanipulationen, Bildschirmattribute setzen, NUMLOCK an, Programmlaufzeit ermitteln, Kommandos ausführen u. a.
DRUCK-BOX Ausgabe von Listen auf Bildschirm, Drucker oder Diskette
FENSTER-BOX Benutzen gerahmter Windows
SUB-BOX Erzeugen und Abarbeiten von Stapelverarbeitungsdateien
SCP-DCP-BOX Datenportabilität zwischen SCP- und DCP-Dateien
USER-BOX Abfragen und Setzen des Userbereiches
ZDIR-BOX Inhaltsverzeichnis einschließlich Subdirectories.

Alle Module wurden unter Turbo-Pascal-Version 3.00 entwickelt; die Nutzung ab Version 4.00 ist möglich.
Steht man vor dem Problem, daß ein Pascal-Programm für unterschiedliche Betriebssysteme oder Hardwarekonfigurationen gepflegt werden soll, so ermöglicht das Programm **TQORG**, daß stets nur eine allgemeine Quelle zu pflegen ist und aus dieser Quelle eine system- oder hardwarespezifische Quelle erzeugt werden kann.
Das Programm **TPASSTRU** ermöglicht die statische und die dynamische Strukturanalyse von Turbo-Pascal-Quellen (ab Version 4.0 werden nicht alle Referenzen gefunden, und die Units können nicht analysiert werden). Das Programm ist ein effektives Mittel für die Dokumentation. Es ist während der Programmier- und Testphase sowie für die Programmwartung nutzbar.
Die beiden Programme stehen für alle Rechner unter SCP, SCP 1700 oder DCP zur Verfügung.

Institut für Forstwissenschaften Eberswalde, Organisations- und Rechenzentrum Potsdam, Virchowstraße 39/41, Potsdam, 1591; Tel. 7 69 41 *Neumann*

256 Farben für EGA- oder CGA-Karte

Es wurde die Unit **COLOR 256** für Turbo-Pascal-Version 5 entwickelt. In ihr werden jeweils zwei Pixel in x-Richtung zusammengefaßt. Somit lassen sich alle benötigten Turbo-5-Grafikbefehle mit folgenden Parametern abarbeiten:
- hochauflösender EGA-Modus:
  Standard: 640 × 350 Pixel,
  16 Farben
  COLOR 256: 320 × 200 Pixel,
  256 Farben
- hochauflösender CGA-Modus:
  Standard: 640 × 200 Pixel,
  2 Farben
  COLOR 256: 320 × 200 Pixel,
  4 Farben.

In Verbindung mit der ebenfalls entwickelten Unit **CGACOLOR** erreicht man im hochauflösenden CGA-Mo-

dus für den Schneider PC 1512 unter Ausnutzung seiner speziellen Eigenschaften folgende Werte:
Standard: 640 × 200
2 Farben
CGACOLOR: 640 × 200
16 Farben
CGACOLOR+COLOR 256:
320 × 200
256 Farben.

Bergakademie Freiberg, Sektion Mathematik, Akademiestraße 6, Freiberg, 9200 *Kraut*

Software für lineare Optimierung

Das Programmsystem **SM – plus** dient zur Verwaltung und zur Lösung linearer Optimierungsprobleme mit der Simplexmethode einschließlich Nachbetrachtung zur Stabilität der optimalen Lösung. Damit können insbesondere linear modellierbare Probleme der Produktionsplanung, der Materialplanung, der Materialausnutzung, Dosierung- bzw. Mischungsverhältnisse sowie Aufgaben der Investitionsplanung rationell bearbeitet werden. Das Programm ist geeignet für alle Betriebsgrößen, insbesondere aber für kleinere und mittelständische Betriebe der metallverarbeitenden Industrie, der Chemie-, Keramik-, Pharma- und Farbindustrie sowie der Landwirtschaft. Das Programm in Turbo-Pascal kann je nach Hardwarevoraussetzung Probleme mit ca. 200 Variablen und 100 Nebenbedingungen, durch zusätzliche Maßnahmen auch noch größere Aufgaben, behandeln.

Technische Universität Karl-Marx-Stadt, Sektion Mathematik, PSF 964, Karl-Marx-Stadt, 9010; Tel. 5 61 21 53 *Würker*

Kopplung Plotter – 8-Bit-Rechner

Zum Anschluß des Plotters **K 6418.02** an die gebräuchlichsten 8-Bit-Rechner wurde ein Treiber mit Ausgabefunktion entwickelt. Die Plotdatei kann unter Nutzung der verschiedensten Texterfassungsprogramme erzeugt werden. Nutzbar sind prinzipiell alle V.24- oder IFSS-Schnittstellen, die das Übertragungsprotokoll DC1/DC3 verwenden. Der entwickelte Treiber arbeitet mit 9600 Baud. Das Treiberprogramm kann zur Anfertigung von Schriftköpfen, Vordrucken, Schönschreibtexten, Beschriftung von Folien etc. genutzt werden. Für folgende Rechnertypen wurden Varianten entwickelt: P 8000, PRG 700, A 5120/K 8924-27, KC 85/2/3, K 8915 und PC 1715.

VEB Metallgußwerk Wernigerode, Abt. TR, Gießerweg 10, Wernigerode, 3700 *Adenstedt*

Baugruppenentwurfssystem

BESSY ist ein CAD-System zur durchgängigen rechnerunterstützten Bearbeitung elektronischer Baugruppen vom ersten Entwurf des Stromlaufplanes bis zur Bereitstellung aller Fertigungsunterlagen. BESSY unterstützt die Bearbeitungsstufen Stromlaufplaneditor, Layouteditor, Postprozeß. BESSY ist erweiterbar, anpaßbar, offen für den technischen Fortschritt, unterstützt fehlerfreie Entwürfe, hat hierarchische Menüs, ist einfach erlernbar durch die klare Bedienerführung, basiert auf einer relationalen Datenbank, unterstützt EGA- und CGA-Grafik, realisiert schnellen Bildaufbau mit einem leistungsfähigen Zoom und maßstäbliche Darstellung. Erforderliche Hard- und Software sind ein IBM-kompatibler PC, 512 KByte Hauptspeicher und mindestens eine 10-MByte-Festplatte, CGA oder EGA, Maus, Matrixdrucker, Plotter SPL-400, MS-DOS (ab Version 3.2 aufwärts).

Hochschule für Seefahrt Warnemünde-Wustrow, Direktorat Wissenschaftlicher Gerätebau, Richard-Wagner-Straße 31, Rostock-Warnemünde, 2530 *Bründel*

EPROM-fähiges Betriebssystem für Turbo-Pascal

Das Betriebssystem **MINITEST** basiert auf dem EPROM-fähigen Turbo-Pascal-Laufzeitmodul, entwickelt vom Institut für Bodenfruchtbarkeit der AdL Müncheberg (MP 9/88). MINITEST wurde für den Microrechner MLW-MR entwickelt (CPU: Z80, SIO, PIO, CTC, EPROM/RAM maximal 64 KByte, K 1520-kompatibler Systembus) und kann durch einfache Modifizierung der Treiber-Routinen aber auch an jeden anderen Z 80-Rechner angepaßt werden. MINITEST läuft unabhängig von handelsüblichen CP/M-kompatiblen Betriebssystemen und ermöglicht ein Programmieren der Nutzerprobleme in Turbo-Pascal 3.0.
Die Kopplung zwischen einem PC 1715 und einem abgesetzten Rechner wird von dem **KOPPEL**-Programm auf dem PC 1715 und dem **MANAGER** des Betriebssystemes MINITEST ermöglicht.

Medizin- und Labortechnik Leipzig, Forschungszentrum Dresden Overbeckstraße 48, Dresden, 8030; Tel. 5 97 31 88. *Landgraf/Sonnhild*

BÜRO

Folgende Softwarelösungen stellen wir Ihnen zur Verfügung: Das Programmpaket **Mail Master** ist ein netzfähiges PC-Programm zur Verwaltung von Kundendaten. Das Besondere sind die Selektions- und Sortiermöglichkeiten.
DAVE verwaltet Ihre Datenträgerbestände und Softwarebibliotheken mit modernem Komfort und bietet Ihnen daneben eine große Palette weiterer Leistungen.
ETI ist ein Programm für Entwurf und Druck von Etiketten, Visitenkarten, Namensschildern u. a. in beliebigem Format.

Benötigen Sie nähere Auskünfte zu unserem Angebot, so können Sie bei uns die Kurzdokumentation anfordern.

Bauakademie der DDR, IPS/Abt. COMPAL, Plauener Straße 163, Berlin, 1092; Tel. 37 83 34 94 *Piálek*

PC 1715-Tastatur mit parallelem Ausgang

Für K 1520 oder ähnliche Systeme mit parallelem Tastatureingang bieten wir die Möglichkeit, eine vorhandene PC 1715-Tastatur umzurüsten oder eine bereits umgebaute Tastatur zu erwerben. Damit können die kaum noch vorhandenen Tastaturen K 7632 ... K 7636 ersetzt werden. Einzige Einschränkung: Ausgaben an die Tastatur sind nicht möglich. Folgende Angebote sind vorhanden: komplett umgebaute Tastatur und Dokumentation, Austauschleiterplatten für vorhandene Tastatur mit Dokumentation, unbestückte Leiterplatte und Dokumentation oder nur Dokumentation.

VEB Robotron Büromaschinenwerk „Ernst Thälmann" Sömmerda, Werk Worbis, Querstraße 2, Wörbis; 5620; Tel. 30 *Becker*

MEOS-Disketten unter CP/M lesen

Bei uns wurde ein Programm geschaffen, mit dem 8"-MEOS-Disketten unter CP/M-80 gelesen und die MEOS-Files als CP/M-Files auf beliebige CP/M-Laufwerke (5,25", 8", RAM-Disk) kopiert werden können. Voraussetzung dafür ist ein CP/M-80-kompatibler Rechner, der 8"-Laufwerke unterstützt. Das sind z. B. umgerüstete MRES und A 51xx. Das Programm **COPYMEOS** ist für das Kopieren von Quelltexten entwickelt worden und führt deshalb keine Konvertierung von verschieblichen oder festen Phasen durch.
Interessenten können das Programm nachnutzen, wenn sie eine Diskette (5,25" oder 8") in einem CP/M-Standardformat einschicken.

Technische Universität Dresden, Sektion Informationstechnik, Bereich Kommunikations- und Computertechnik, Mommsenstraße 13, Dresden, 8027; Tel. 4 63 39 41 *Johannsen*

Raumplanung

Es wurde ein universell einsetzbares Programm zur Verwaltung umfangreicher Raumpläne, z. B. von Universitäten, Schulen, Ausstellungs- und Messezentren u. ä., für den Arbeitsplatzcomputer A 7100 entwickelt. Behandelt werden regelmäßige (wöchentlich und 14täglich) sowie einmalige Veranstaltungen in maximal 5 Gebäuden zu je 255 Räumen. Das Programm bietet die Möglichkeit, nach Teilnehmern, Veranstaltern oder freien Räumen zu suchen, sowie die des Druckes von Stunden- und Raumplänen. Bei der Suche nach freien Räumen wird nach Kapazität und technischer Ausstattung unterschieden und der ausgewählte Raum mit einer gewünschten Veranstaltung belegt.
Bei Zusendung einer Diskette bieten wir jedem Interessenten die Dokumentation des Programms.

Karl-Marx-Universität Leipzig, Sektion Mathematik, Karl-Marx-Platz, Leipzig 7010 *Gumpert/Fischer*

Verschlüsselung von Zeichnungsnummern

Das Programmsystem **CODE35** nimmt Verschlüsselungen zu 8stelligen Kodes vor. Der Kode kann unter MS-DOS als Zeichnungsname Verwendung finden. Damit besteht ein direkter Zusammenhang zwischen Zeichnungsfilenamen und Zeichnungsnummern. Es ist eine Lösung für folgende Zeichnungsnummernstruktur vorhanden: xxx.xx-xxx.x/x
Änderungen des Programms sind für ähnliche Strukturen möglich. Funktionen: Kodierung, Dekodierung, Ausgabe einer Tabelle tl. verschlüsseltem Directory mit Dekodierung, Ausgabe einer Tabelle von Teilen des Directorys (Zeitraum, bestimmte Extension, bestimmte Zeichnungsnummer (von–bis)), Suchen eines Zeichnungsfiles auf einem verschlüsselten Directory, Erzeugen des Zeichnungsfiles für PolyCAD.

VEB Polygraph Druckmaschinenwerk PLAMAG, Abt. TS, CAD-Gruppe, Postfach 47, Pausaer Straße 284, Plauen, 9900; Tel. 28 14 13 *Streich*

Programmpaket TASTAR

Das Programmpaket TASTAR ermöglicht es, Programme zur Datenerfassung und Datenpflege im Dialogbetrieb zu erzeugen. Die hiermit bearbeiteten Dateien sind die Dateien des aus Turbo-Pascal bekannten Turbo-Access-Systems. Diese ermöglicht den Direktzugriff auf jeden Datensatz anhand seines Schlüssels. Darüber hinaus besteht die Möglichkeit, Textdateien aus diesem System auszugeben.
Zum Programmpaket gehören die drei Programme ETAF (Entwurf TA-File) und UTAF (Umsetzung TA-File) sowie die Quelltextkomponente BTAF (Bearbeiten TA-File). Das Programmpaket TASTAR gibt es für 8- und 16-Bit-Technik.

VEB Chemiefaserwerk Premnitz, Abt 252 (ORZ), Premnitz, 1832; Tel. 4 34 63 *Daur/Wiesener*

Leiterplattenkapazität

Es wird freie Kapazität für durchkontaktierte Leiterplatten einschließlich Trockenstrecke angeboten. Folgende Arbeiten werden durchgeführt:
1. Layout-Digitalisierung (Herstellung von Bohrlochstreifen und Lichtzeichenunterlagen)
2. Leiterplatten bohren laut Bohrlochstreifen
3. Planfilm/Glasoriginal/Lichtzeichnen
4. Entwicklung Planfilm/Glasoriginal
5. Herstellung Planfilm für Lötmasken und Bestückungsaufdruck, unter den Voraussetzungen:
1. Maßhaltiges Layout, Maßstäbe 1:1, 2:1, 4:1
2. Bohrplan mit Lochdurchmessern und Lochanzahl pro Durchmesser nach TGL 31431, Bl. 1
3. Stromlaufplan nach TGL 16084, Bl. 1
4. Bestückungsplan nach TGL 16085, Bl. 1.

VEB Betonwerke Laußig, Werk Kemberg, Straße der MTS 2, Kemberg, 4604 *Krüger*

Wir suchen ...

... Konvertierungsprogramme für CADy ab Version 3.04, PC-LES und LKA85 zur Ansteuerung der ADMAP 5. mit einem 8-Bit-Bürocomputer

INTERFLUG, Flugversicherung/RMEB, Flughafen, Berlin, 1189; Tel. 6 72 77 31/75 92/77 05 *Spangenberg/Grabow*

80386 SX-Familie jetzt auch mit 20 MHz

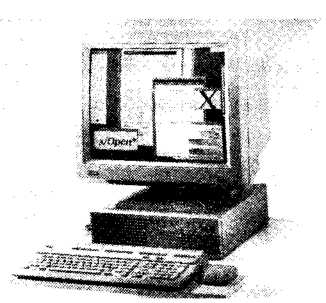
Während einige Hersteller Versionen des 16-Bit-Prozessors 80286 bereits mit 20 und 25 MHz liefern, war die Taktfrequenz von Intels 80386 SX bisher auf 16 MHz begrenzt. Nun beginnt Intel mit der Lieferung auch einer 20-MHz-Version des 80386 SX. Ein System auf der Basis des 20-MHz-386 SX und mit dem Cache-Controller 82385 SX-20 soll dann gegenüber auf dem Mikroprozessor 386 SX-16 basierenden Systemen Leistungssteigerungen von bis zu 40 Prozent erzielen.
Der Mikroprozessor 386 SX ist eine Version des Industriestandards 386 DX mit externem 16-Bit-Bus. Diese Eigenschaft erlaubt es Computerherstellern, Produkte mit den Kompatibilitäts-, Multitasking- und Leistungsvorteilen der Intel-386-Architektur zu Preisen vergleichbar ausgestatteter 16-Bit-Systeme anzubieten.
Die 20-MHz-386 SX-Familie umfaßt:
- Mikroprozessor 386 SX
- PC AT-Chipsatz 82340 SX
- Cache-Controller 82385 SX
- Arithmetik-Koprozessor 387 SX
- LAN-Koprozessor 82596 SX.

Die neue 20-MHz-Version wird in Intels CHMOS-IV-Prozeßtechnologie gefertigt, welche die Leistungsaufnahme gegenüber Produkten gleicher Taktfrequenz – produziert mit den Vorgänger-Prozeß – um über 30 Prozent reduziert. CHMOS IV erweitert den zulässigen Betriebstemperaturbereich von 0 bis 85 °C und steigert die Zuverlässigkeit des Bauelements.
Für Notizbuch- und Laptop-PCs bietet Intel eine Version mit noch geringerer Verlustleistung an, die mit reduzierter Frequenz von 2 MHz arbeitet, wenn das System in einen Standby-Mode mit niedriger Leistungsaufnahme versetzt wird.
Für PC AT-Entwicklungen ist der *Chipsatz 82340 SX* verfügbar. Dieser neue Zwei-Chip-Satz bietet den OEMs die höchstintegrierte Buslösung der Industrie. Er ermöglicht den Entwicklern den Aufbau von SX-Systemen mit weniger Chips – keine TTL-Schaltungen erforderlich – für billigere und kompaktere Modelle.
Durch den Einsatz des *Cache-Controllers 82385 SX-20* kann eine Leistungssteigerung von bis zu 40 Prozent erzielt werden. Der Controller erreicht dies durch Reduzierung der Prozessor-Wartezyklen und der Bus-Zugriffszeiten. Für hohe Systemflexibilität unterstützt der 82385 SX-20 sowohl direkt abbildende als auch zweiweg-assoziative 16-KByte-Cachedesigns. Darüber hinaus integriert er das Cache-Directory und die Cacheverwaltungslogik auf einem einzigen Chip und unterstützt den vollen 386 SX-Adreßraum von 4 GByte.
Eine 20-MHz-Version des *Arithmetik-Koprozessors 387* zur Unterstützung des 386 SX-20 ist ebenfalls von Intel verfügbar.
Der *32-Bit-LAN-Koprozessor 82596 SX-20* arbeitet parallel zum 20-MHz-Mikroprozessor und übernimmt unabhängig davon alle LAN-Funktionen in CSMA/CD-Netzwerken. Der 82596 SX überträgt Daten mit der vollen Bandbreite des 16-Bit-E/A-Systembusses und der 10-Mbit/s-Ethernet-LANs und benötigt zur Unterstützung eines voll ausgelasteten Ethernet-Netzwerks lediglich sechs Prozent der Systembusleistung.
Der LAN-Koprozessor läßt sich auf der Systemplatine ohne zusätzliche externe Beschaltung unterbringen und reduziert damit die Bauteile-Anzahl, die Komplexität und die Kosten des Systems. *MP*

Keine Weiterentwicklung von Turbo-Basic

Borland wird Turbo-Basic nicht mehr weiterentwickeln; die Borland GmbH Deutschland bietet allen Turbo-Basic-Kunden ein preiswertes Update anderer Programmiersprachen, beispielsweise Turbo-Pascal oder Turbo-C, auch in deren Professional-Versionen an. Turbo-Basic wird zwar weiterhin von Borland betreut; die Weiterentwicklung wird jedoch in den USA von dem Unternehmen „Spectra Software" übernommen. Bereits im Dezember soll die erste Version des nun PowerBasic genannten Produkts veröffentlicht worden sein. *MP*

Sharp nun auch mit MS-DOS-Laptop



Als sensationell bezeichnet Sharp seinen zur CeBIT '90 vorgestellten Mini-Laptop-Computer. Der netzunabhängige **PC-6220** wiegt unter 2 kg, kommt im A4-Format und ist nur 35 mm dünn: Ein extrem leichtgewichtiger und dabei vollwertiger PC, der – in einem Komponentensystem – als echter Arbeitsplatz-PC einsetzbar sein soll. Er ist ausgestattet mit einem wahlweise 6, 8 oder 12 MHz schnellen 80C286-Prozessor, 1 MByte RAM (aufrüstbar auf 3 MByte), einer 2,5-Zoll-Festplatte mit 20 MByte und 23 ms mittlerer Zugriffszeit. Für Erweiterungen steht ein Sockel für einen 80C287-Koprozessor zur Verfügung. Der PC-6220 enthält das Betriebssystem MS-DOS 4.01 im ROM und verfügt unter anderem über eine serielle und eine parallele Schnittstelle.
Etwas Besonderes ist der Bildschirm: Hier setzt Sharp erstmals die neue Triple-Supertwisted-Technik (TST) ein, die eine VGA-kompatible Darstellung mit 640 × 480 Bildpunkten schwarz auf weiß ermöglicht. Gegenüber herkömmlichen LCDs werden noch höherer Kontrast und mehr Graubabstufungen bei leichterer und schmalerer Bauweise erreicht.
Eine weitere Besonderheit ist das Komponenten-System mit einem für derartige Laptops neuen Erweiterungskonzept. Über Steckanschlüsse ist der PC-6220 durch einfaches Anfügen weiterer Komponenten zu einem kompletten netzwerkfähigen PC-System aufrüstbar. Zu den Komponenten gehören eine 10er-Block-Tastatur, ein 3,5-Zoll-Diskettenlaufwerk und eine Expansionsbox, die zwei freie Steckplätze für AT-kompatible Karten enthält. Für den Datenverkehr zu anderen Computern ist die Kommunikationssoftware im Lieferumfang enthalten. In den Handel kommt der PC-6220 ab Mitte des Jahres, der noch nicht feststehende Preis soll unter 10 000 DM liegen. *MP*

Microsoft: Entwicklungsumgebung für Assembler

Quick-C wurde jetzt von Microsoft um einen Quick-Assembler erweitert, der in die Entwicklungsumgebung von Quick-C integriert wurde. Damit werden sich die beiden – nach Marktuntersuchungen von Microsoft – am häufigsten von Entwicklern auf PCs benutzten Sprachen in Zukunft in einer gemeinsamen, einheitlichen Bedienoberfläche präsentieren. Die zeitkritischen Anwendungen und Zugriffe auf die Systemebene von MS-DOS oder auf die Hardware in Assembler-Code können jetzt wesentlich einfacher in den C-Code eingebunden werden. In dem Paket enthalten sind ein Editor, Compiler und Debugger. Sie werden aber gegenwärtig nur in einer englischen Version ausgeliefert. Die meisten Entwickler dürfte dies aber kaum am Einsatz hindern oder gar abschrecken. *MP*

Neue Unix-Workstation Siemens WX200

Siemens erweitert seine Arbeitsplatzsysteme um die in Augsburg gefertigte Unix-Workstation-Familie WX200, die eine volle Sinix-Integration garantiert.
Sie setzt sich aus vier Modellen zusammen: zwei Desktop-Varianten (10DT und 20DT), von denen eine das Bild zeigt, und zwei Tower-Modellen (30T und 40T). Eingesetzt werden die Intel-Standardprozessoren 386/486, die modellabhängig von 16 MHz aufwärts bis zu 33 MHz getaktet sind. Der Anwender hat die Wahl zwischen Monochrom- und Farb-Bildschirmen. Der kleine Bildschirm bietet eine Auflösung von 640 × 480 Pixeln, die größeren bieten 1280 × 1024 Bildpunkte. Beim 19-Zoll-Farbbildschirm können 256 Farben gleichzeitig dargestellt werden.
Als Software wird Sinix Open Desktop angeboten, das auf dem anerkannten Open Desktop – mit Unix System V/386 Release 3.2 – der Santa Cruz Operation (SCO) basiert. Dieses Softwarepaket ist eine integrierte Betriebssystemumgebung, die alle Komponenten vom Betriebssystem über Datenhaltung und Vernetzung bis hin zur grafischen Bedienoberfläche umfaßt. Zusätzlich zur weitreichenden Quellcode-Kompatibilität innerhalb der Unix-Welt (z. B. X/Open, Unix V.2 und V.3) wird als Besonderheit Binärcodekompatibilität zu anderen Rechnern mit Intel-Prozessoren (z. B. mit Open Desktop oder Unix System V/386) und damit der direkte Zugang zu einem weltweiten Softwaremarkt geboten. Daten und Programme innerhalb der WX200-Familie können problemlos ausgetauscht werden.
Die Workstation bietet mit Sinix Open Desktop:
- Unix nach internationalen Standards (X/Open)
- grafische Bedienoberfläche OSF/MOTIF, basierend auf X/Window
- relationales Datenbanksystem mit SQL
- volle LAN-Anbindung (z. B. TCP/IP, NFS)
- DOS-Emulation unter Unix; damit können Unix- und MS/DOS-Programme gleichzeitig genutzt werden.

Siemens hat das Open Desktop von SCO um X/Open-Funktionen ergänzt und wird dieses integrierte Softwarepaket bei X/Open zur Validierung vorlegen.
Haupteinsatzbereiche der WX200 liegen im Büro mit Sinix-Standardanwendungen und Desktop-Publishing, in Aufgabengebieten des 2D-CAD (Variantenkonstruktion, Stromlaufpläne) mit Sigraph, beim Software-Engineering (CASE) mit Domino und im Feld der Expertensysteme, wo zum Beispiel mit Prolog ein geeignetes Werkzeug für die Entwicklung solcher Systeme zur Verfügung steht. *MP*

Neuer Motorola 68040 im Einsatz

Von zahlreichen namhaften Computerherstellern liegen bereits Absichtserklärungen vor, das neue Flaggschiff der Motorola-Prozessoren, den 68040, als Grundlage für Computerentwicklungen zu nutzen.
Erster scheint nun die britische Firma Tadpole Technology geworden zu sein. Sie stellte zwei Rechner vor: den TP 40V speziell für Netzwerke und den TP 41V mit VME/VSB-Bus. Die Systeme besitzen je eine Ethernet-Schnittstelle und zwei RS-232-Interfaces, letzteres zusätzlich eine SCSI-Schnittstelle sowie 32 MByte maximalen Hauptspeicher. Die Leistung der Systeme soll bei 20 MIPS bzw. 3,5 MFLOPS liegen.
Als Betriebssystem wird eine Version des AT & T-Unix-Systems V3 angeboten. *MP*

Miniatur-Diskette

Die Firma IBM gab die Entwicklung einer sechs Quadratzentimeter großen Diskette bekannt. Auf dieser Diskette sollen Informationen von zirka 100000 A4-Seiten untergebracht werden können. Das entspräche einer Speicherkapazität von einer Milliarde Bit. Um diese Gigabit-Speicherdichte zu erreichen, seien neuartige Komponenten, darunter Dünnfilm-Speicherköpfe und -platten, sowie hochentwickelte Elektronik kombiniert worden. Bei dem Prototyp sollen alle kritischen Hardware-Komponenten in konventionellen Fertigungsprozessen hergestellt worden sein. Der experimentelle, aus zwei Elementen bestehende, Dünnfilm-Speicherkopf enthält laut IBM ein induktives Schreibelement und ein magnetoresistives Leseelement. Er fliegt in einer Höhe von weniger als 0,05 μm über die Plattenoberfläche. Die Speicherköpfe gegenwärtig erhältlicher Plattenlaufwerke gleiten in einer Höhe von 0,15 bis 0,38 μm über die Plattenoberfläche.
Die neuentwickelte Diskette besteht aus Aluminium, das mit einer Kobaltlegierung überzogen wurde. Ein dünner Überzug aus hartem Material schützt die Legierung vor einem Kontakt mit dem Speicherkopf.
Bevor die neuen Disketten im Handel erhältlich sind, sollen nach Angaben von IBM noch mehrere Jahre vergehen.

Quellen: Handelsblatt vom 2. 1. 1990 Eildienst. – Berlin (1990) 17. – S. 4 *Fa*

Speicherelemente dichter packen

Um die Packungsdichte bei elektronischen Speicherelementen erheblich zu erhöhen, arbeitet eine Gruppe von Elektronikern des japanischen Unternehmens Sharp Corp., Osaka, an der Entwicklung neuer Werkstoffe. Wenn es möglich würde, die Isolationsschicht zwischen den Elektroden von Kondensatoren zu reduzieren, würde sich der Platzbedarf für die Kondensatoren dynamischer Speicherchips um rund 20 % und beim kompletten Chip insgesamt um rund 10 % verringern. Mit den gegenwärtig benutzten Werkstoffen ist das wegen der zu hohen Leckströme nicht möglich.
Als Isolationsschicht testet die japanische Entwicklergruppe Siliziumnitrid, das eine verbesserte elektronische Isolationswirkung aufweist und lediglich noch eine Dicke von 3,8 nm benötigt. Für die Fertigungstechnik ergeben sich damit geringe Veränderungen.
Die Firma ist jetzt dabei, das neue Verfahren für die Produktion von 16-MBit-DRAMS aufzubauen.

Quelle: Blick durch die Wirtschaft vom 18. 12. 1989 *Fa*

Hochintegrierte schnelle optoelektronische Chips für Großrechner

Einer Gruppe von Wissenschaftlern der Firma IBM ist es gelungen, einen Bausatz zu entwickeln und herzustellen, der aus zwei optoelektronischen Chips für Hochgeschwindigkeitsverbindungen in Computern besteht. Dieser Bausatz, der aus einem Sender- und einem Empfänger-Chip besteht, soll Datenraten bis zu 1 GBit/s übertragen können.
In Bereichen mit rechenintensiven Computeranwendungen, wie Wettervorhersagen und Flugsimulation, müssen Milliarden und teilweise sogar Billionen von Datenbits verarbeitet werden. Die Übertragung derart umfangreicher Datenmengen von einem Rechner zum anderen erfordert effizientere und leistungsfähigere Transporttechniken.
Muster des optoelektronischen Sender-Chips bestehen aus vier runden Stäben, die in ihren Aussparungen optische Fasern von der Stärke eines menschlichen Haares haben. Auf dem Rand vor jeder Faser sind vier leistungsstarke, mikroskopisch kleine Laser zu erkennen, die mit einer elektrischen Leitung verbunden sind. Sie wandeln die digitalen Computersignale in Lichtimpulse um. Die optischen Fasern sammeln das Licht jedes Lasers und senden es zu anderen Rechnern.
Auf dem Empfänger-Chip befinden sich 50mal mehr optische und elektronische Bauelemente, als bisher jemals auf einem optoelektronischen Chip integriert wurden, darunter über 8000 Transistoren mit Strukturbreiten von 1 μm. Die optischen Fasern werden in einem genau berechneten Winkel schräg angeschnitten, so daß das zu übertragende Lichtsignal gezielt auf die Fotodetektoren auftrifft. Diese sind nicht zu erkennen, da sie im Schattenbereich der Fasern liegen. Die Fotodetektoren nehmen die Lichtimpulse, die mit einer Geschwindigkeit von 1 GBit/s ankommen, auf und wandeln sie in elektrische Signale um. Sender- und Empfänger-Chip werden aus GaAs hergestellt, das sowohl für sehr kleine Laser und Lichtdetektoren als auch für sehr schnelle elektronische Schaltkreise mit geringem Energieverbrauch geeignet ist. Auf den Chips wird eine große Anzahl der elektronischen Schaltkreise mit den lichtaussendenden und lichtempfangenden Bauelementen verbunden, so daß die in MESFET-Technik (Metal semiconductor field effect transistor) entwickelten Chips sowohl digitale und analoge als auch optische Funktionen umfassen. Dadurch wird eine wesentliche Steigerung der Leistungsfähigkeit in der Datenkommunikation erzielt. Der der Datenübertragung dienende Laserstrahl wird von einem weiterentwickelten GaAs-Injektionslaser erzeugt.
Die bisher nur experimentell hergestellten Muster-Chips sind bereits als fertige Bauelemente montiert worden, wobei Übertragungsleistungen von 1 GBit/s erreicht wurden. Über Vorbereitungen für den industriellen Einsatz erfolgten noch keine Veröffentlichungen.

Quelle: Mikroelektronik. – Berlin(W)/ Offenbach (1989) 6 *Fa*

Computer mit künstlicher Intelligenz

Das VW-Tochterunternehmen „Gesellschaft für technische Datenverarbeitungssysteme" (VW-GEDAS) will nach eigenen Angaben mit Ergebnissen aus staatlich finanzierten Forschungsaufträgen zum Marktführer im Bereich der Computer mit Künstlicher Intelligenz werden. Dazu soll das von der Gesellschaft für Mathematik und Datenverarbeitung (GMD) entwickelte System BABYLON zu einem industriegerechten Produkt weiterentwickelt und professionell vermarktet werden (Entwicklungszeit: 18 Monate). VW-GEDAS will bis 1995 rund 10 Mrd. DM investieren, um BABYLON zu einem leistungsfähigen „Expertensystem zur Produktion von Expertensystemen" für alle Computertypen (vom PC bis zum Großrechner) weiter entwickeln und vermarkten zu können. *Fa*

Chip-Entwicklung bei IBM

In dem neuen Semiconductor Technology Center (ASTC) der Firma IBM sollen künftig Logik- und Speicherchips mit Hilfe der Röntgenstrahllithografie entwickelt und auch für die Serienfertigung vorbereitet werden. Im ASTC wird der derzeit einzige im nichtstaatlichen Besitz befindliche Synchrotronspeicherring der USA für Röntgenstrahllithografie installiert. Diese Technologie ermöglicht es, die Grenzen der momentan eingesetzten optischen Lithografie zu unterscheiden. Damit lassen sich Strukturen von weniger als 0,5 μm Breite auf Silizium abbilden, und die Herstellung von noch dichteren, schnelleren und leistungsfähigeren Chips wird möglich.
IBM beabsichtigt, mit der US-Regierung zusammenzuarbeiten. Sie machte das Angebot, ihr Röntgenstrahl-Laboratorium mit ausgewählten Firmen gemeinsam zu nutzen. Ein erster Kooperationsvertrag wurde bereits mit dem Chipproduzenten Motorola abgeschlossen. Dieser Vertrag sieht vor, daß sechs bis acht Ingenieure der Firma Motorola zunächst für 21 Monate am Röntgenstrahllithografie-Programm von IBM mitarbeiten.
IBM investierte bereits rund eine halbe Milliarde Dollar in das ASTC, mit dessen Bau vor drei Jahren begonnen wurde. Die endgültige Inbetriebnahme ist für 1991 vorgesehen.

Quelle: Konstruktion und Elektronik. – Landsberg (1989) 51 – S. 1 *Fa*

Schnelles Text-Datenbanksystem

Zum elektronischen Einlesen und Wiederauffinden von Texten hat die Firma Hitachi Ltd. ein sehr schnelles Datenbanksystem entwickelt. Dabei wird jedes Wort, das in einem Text enthalten ist, als Suchbegriff genutzt. Ohne aufwendiges Indizieren wie bei herkömmlichen Datenbanksystemen kann jedes Wort zum direkten Suchen in den gespeicherten Texten verwendet werden. Ein solches Suchen nach gewünschten Informationen dauert bei einer Wortmenge, die der eines Zeitungsjahres (rund 72 Millionen Wörter) entspricht, nur etwa 1,5 Sekunden.
Bei der Volltextrecherche (dreistufige Recherche) werden automatisch anhand eines vorgegebenen Suchbegriffes Synonyme und Wörter generiert, die denselben Sachverhalt beschreiben. Ein hierarchisches Vorauswahlsystem bearbeitet automatisch eine Zusammenstellung verdichteter Texte und eine Zeichenzusammensetzungstabelle vor den komprimierten Informationen. Das gesamte System umfaßt:
– eine RAM-Floppy mit einer maximalen Lesegeschwindigkeit von 20 MByte/s
– ein Magnetplattensystem mit 10 MByte/s
– einen Suchprozessor mit einem Durchsatz von 1,6 MByte/s
– einen spaltenorientierten Zeichensuchprozessor mit 20 MByte/s.
Das entspricht insgesamt einer Systemantwortgeschwindigkeit von 100 MByte/s. 25000 technische Dokumente mit durchschnittlich je 10000 Wörtern können in rund fünf Sekunden durchsucht werden.
Das System eignet sich zur Informationsverwaltung gespeicherter Patente, Bücher sowie Zeitungs- und Zeitschriftenartikel.

Quelle: New Technology Japan. – (1989) 11 *Fa*

NEC mit 64-MBit-Chip-Technik

Die Firmen NEC Corp. und Sanyo Chemical Industries Ltd. haben gemeinsam ein organisches Polymer aus Styrol und Chlormethylstyrol zur Herstellung eines „Mikrochips mit flacher Oberfläche" entwickelt. Mit dieser Technik könnten Halbleiter produziert werden, die eine Speicherkapazität von 64 MBit oder mehr aufweisen. Diese neue Technik kann bei CCDS, LCI sowie bei Magnetköpfen für dünne Filme eingesetzt werden.

Quelle: Blick durch die Wirtschaft vom 9. 1. 1990 *Fa*

Dreidimensionale Bauelementeanordnung auf Leiterplatten

Die Erhöhung des Integrationsgrades und der Komplexität der Schaltungen führt zu einer gesteigerten Anzahl von Verbindungen der Chips untereinander. Trotz der Verringerung der Leiterbahnbreite sind die Möglichkeiten zur Unterbringung auf der zweidimensionalen Leiterplattenfläche nicht mehr ausreichend, weshalb die dritte Dimension genutzt wird.
Unter der Bezeichnung ‚Chiprack' wird von der Firma Ultra Electronics eine Lösung vertrieben, die eine weitere Erhöhung der Packungsdichte gestattet. Sie besteht aus Bauelementefassungen und Verbindungselementen. Die Fassungen sind doppelseitig mit Kontakten versehen und führen Verbindungen nach oben und nach unten. In dieser „Regalanordnung" bestehen Verbindungen von einer Ebene zur anderen. Die reguläre Anordnung aller Blöcke und Stapel ermöglicht eine einfache Montage. An der Außenseite jedes Stapels befinden sich außerdem Anschlüsse, die für Tests oder zusätzliche Verbindungswege genutzt werden können.
Ein kleines Prototypsystem – auf dem Prozessor Z-80 basierend – benötigt nur 20 % des Volumens gegenüber konventioneller Verbindungstechnik. Ein in SMD-Bauweise ausgeführtes System benötigt mehr als das doppelte Volumen des Chiprack-Designs.

Quelle: eee. – Leinfelden-Echterdingen (1989) 25. – S. 52, 53 *Fa*

Einführung in die digitale Vermittlungstechnik

von H. Klier u. V. Seidel, Reihe Informationselektronik, VEB Verlag Technik, Berlin 1989; 184 S., 123 Abbildungen, 15 Tafeln, DDR 19,00 M

Fachliteratur zur Vermittlungstechnik auf unserem Büchermarkt entspricht dem Stand der Vermittlungssysteme in unseren Netzen. Diese merkliche Lücke können auch nicht Informationen in Fachzeitschriften oder in der Reihe „Berichte zur Nachrichtentechnik" schließen. Um so mehr Anerkennung verdient die Leistung der Autoren. Viele Ingenieure und Techniker der Fernmeldetechnik, der Hard- und Software, Studenten und sicher in Kürze sogar Lehrlinge und Facharbeiter werden dafür sorgen, daß der Titel rasch vergriffen ist. Man muß berücksichtigen, hier handelt es sich um eine spezielle Einführung in High-Tech.

Das Anliegen der Autoren, in knapper Form, den roten Faden nicht aus den Augen verlierend, den Gegenstand so vollständig zu beschreiben, daß dem Leser ein Grundwissen vermittelt wird, auf dem er aufbauen kann und das den Einstieg in weiterführende Literatur (vgl. auch Quellenverzeichnis) erleichtert, ist gut gelungen. Ergänzt sei, daß aufgrund der immer größer werdenden Breite der Nachrichtentechnik sich jeder Spezialist bei Bedarf schnell über neue Beschreibungsmethoden und Funktionsprinzipe der digitalen Vermittlungstechnik orientieren kann. Die ersten 20 Seiten sind als Grundlage für das weitere Verständnis der PCM-Technik gewidmet. Der Abschnitt 3 zu den Aufgaben der Vermittlungstechnik beinhaltet das grundsätzliche Einordnen der Vermittlungsstellen in das Netz und die Charakteristika der wesentlichen Blöcke. Abschnitt 4 nutzt moderne Darstellungs- und Beschreibungsformen. Das Einarbeiten in SDL (Specification and Description Language) bereitet dem rechentechnisch vorgebildeten Leser keine Probleme. Der Vermittlungstechniker, der noch an Stromlaufplan und Relaisdiagramm gewöhnt ist, sollte diesem Punkt mehr Aufmerksamkeit widmen oder noch zusätzliche Literatur zu Rate ziehen. Im 5. Abschnitt werden Koppelanordnungen und Steuerungsprinzipe beschrieben und verglichen. Der folgende Abschnitt untersetzt das nochmal mit möglichen Bauelementetechnologien.

Wenn man berücksichtigt, daß die Software für ein Vermittlungssystem international über 30 Prozent bei der Entwicklung in Anspruch nimmt und für den Betrieb erheblich ist, ist es logisch, daß die Autoren im 7. Abschnitt dafür 40 Seiten verwenden. Hier wird das notwendige Umlernen für die Fachleute der konventionellen Vermittlungstechnik nochmals ganz deutlich. Der knappe Ausblick auf das ISDN erscheint aus der aktuellen Situation heraus als ausreichend.

Bei diesem Buch ist man sich jederzeit sicher, daß die Autoren wissen, wovon sie schreiben; sie bringen auch Entwicklungen mit ein, an denen sie selbst beteiligt waren. Nur dem aufmerksamen Leser oder dem Fachmann fallen einige „Schönheitsfehler" auf. Sie betreffen die Terminologie (Jargon, zweideutiger Terminigebrauch, Nichtbeachtung jüngster Standards), vereinzelte Auswahl untypischer Beispiele zur Beweisführung, wenige Schreib- und Interpunktionsfehler.

Letzteres und das kaum hilfreiche Sachwortregister sind sicher dem Lektor anzulasten.

Wenn man ins Kalkül zieht, welche Entwicklung unsere öffentlichen und nichtöffentlichen Netze und Dienste nehmen müssen, ist das vorliegende Buch ein willkommener, guter Einstieg. Ausbildungs-, wissenschaftliche, praxisorientierte Literatur und Nachschlagewerke müssen folgen.

Bergmann

Rationelle Textverarbeitung mit WordPerfect 5.0

von H.-P. Förster u. M. Zwernemann, Reihe Chip Wissen, Vogel Buchverlag, Würzburg 1989, 160 S., 37 Abb., 2 Disketten
5,25 Zoll, 48,– DM

Wer Gelegenheit hatte, das enorm mächtige Textverarbeitungssystem WordPerfect 5.0 und dessen Leistungsspektrum einmal aus der Nähe zu begutachten, der wird gut verstehen, wie dankbar jeder Anwender über verständlich und übersichtlich geschriebene Literatur sein muß, die den hoffnungslos überforderten Original-Handbüchern zur Seite stehen will.

Ein Anspruch, dem das vorliegende Buch durchaus gerecht wird. Das ist wohl vorrangig auf die Tatsache zurückzuführen, daß sich dieses Buch auf ausgewählte Probleme beschränkt, die jeder anspruchsvolle Anwender eines jeden Textverarbeitungssystems schon irgendwie bewältigt haben möchte. Dies muß gar nicht in jedem Fall etwas mit den spektakulären Möglichkeiten der Grafikeinbindung von WordPerfect zu tun haben. Hier ist in der Praxis in den allerwenigsten Fällen etwas falsch zu machen. Wie wäre es aber zum Beispiel mit einem stilgerechten und professionellen Layout der Texte per Tastendruck? Ein durchaus nicht einfaches Problem, wenn es wie bisher für jeden Text auch noch mühselig neu entworfen werden muß. Wenn man dann auch noch mit Hilfe von „Textbausteinen" seine Schreibarbeiten rationalisieren will, steht man scheinbar vor einem hoffnungslosen Dilemma.

An dieser Stelle springen die ersten Kapitel dieses Buches ein und führen überzeugend und an vielen Beispielen die Vorzüge dieses Textverarbeitungssystems vor.

Aber auch die folgenden Kapitel, die sich eingehend mit den Misch- und Rechenfunktionen von WordPerfect beschäftigen und Leistungsfähigkeit dokumentieren, indem Briefe gleich serienweise ihre Adressen aus dBase-Dateien holen und Rechnungsformulare sich praktisch selbst erstellen, lassen den Leser kaum lokker und regen zum Nach- und Bessermachen an. Wenn dann allerdings versucht wird, die ohnehin schon streitbare «Marktprogrammiersprache» von WordPerfect umfassend zu erläutern, wird auch dem interessierten Leser eine ziemliche Durststrecke zugemutet. Die Zielgruppe dieser Sprache und ebenso dieser Lektion ist mir nicht klargeworden: Denn was den „Textverarbeiter" mit großer Wahrscheinlichkeit überfordert, das frustiert den „Programmierer" ob seiner kläglichen Möglichkeiten sicherlich ebenso.

An dieser Stelle sei mir auch die Frage gestattet, warum dem Leser mit den Abbildungen, die den Bildschirminhalt wiedergeben sollen, ein Grafikmodus – und auch noch in profaner CGA-Qualität – suggeriert wird? Ein wesentlicher Vorteil von WordPerfect besteht für mich doch gerade darin, daß es weitgehend alle Bildschirmaktivitäten im technisch einwandfreien Textmodus abhandelt. Selbst eingebundene Grafiken werden während der üblichen Schreibarbeit als leere Kästchen dargestellt, und erst bei Bedarf ist man z. B. mit dem sogenannten Page-preview-Modus in der Lage, sich die entsprechenden Abschnitte in der endgültigen Fassung und im Grafik-Modus anzusehen. Dies gewährleistet zügiges Arbeiten und schont zudem die Augen.

Selbstverständlich tut dies der inhaltlich gut aufgearbeiteten und ohne Zweifel nützlichen Lektüre keinen nennenswerten Abbruch. Die zwei zugehörigen Disketten erleichtern nicht nur über die mitgelieferten Übungstexte das Nachempfinden der Probleme und Lösungen, sondern sie bereichern die „hauseigene" Grafikbibliothek auch noch mit aussagekräftigen Piktogrammen.

Ein insgesamt empfehlenswertes Buch für alle, die das moderne und erfolgreiche Textverarbeitungssystem über den gelegentlichen Gebrauch hinaus nutzen wollen. WordPerfect-Vorkenntnisse sollten aber dringlich vorhanden sein.

B. Prager

Integrierte Software im Vergleich

von E. Hölzenbein und A. Krings, Dr. Alfred Hüthig Verlag Heidelberg, 1989

Gerade auf dem Gebiet der Büroautomatisierung haben durch den breiten Einsatz leistungsfähiger PC-Technik integrierte Softwaresysteme stark an Bedeutung gewonnen. Trotzdem sind ihre konkreten Einsatzmöglichkeiten nach wie vor weitgehend unklar. Speziell zwei Fragen wurden bisher nicht geklärt:

1. Welches integrierte System ist für welche Aufgabenstellung am besten geeignet?
2. Für welche Aufgabenstellungen ist der Einsatz eines integrierten Systems statt herkömmlicher Einzelsysteme (z. B. dBASE-III oder WordStar) sinnvoll?

Das vorliegende Buch versucht durch einen Vergleich verschiedener Systeme zur ersten Frage Antworten zu geben. Die zweite Frage wird nicht berührt. Beim ersten Lesen des Buches fällt die klare, übersichtliche Gliederung und die gute Strukturierung auf. Abbildungen von konkreten Bildschirmausgaben bzw. Menüs erleichtern das Verständnis. Besonders positiv ist die Zusammenfassung der Ergebnisse in tabellarischer Form zu bewerten. Das Buch scheint tatsächlich einen guten Überblick über die einzelnen Systeme zu geben.

Beim genauen Studium fallen jedoch eine ganze Reihe von Schwächen auf. Zunächst ist die Auswahl der zu vergleichenden Systeme etwas unverständlich. Während von den sechs integrierten Systemen vier auf IBM-PCs unter MS-DOS laufen, nämlich FRAMEWORK II, OPEN ACCESS II, Office Manager und Symphony, laufen die Systeme Jazz und Ragtime auf dem Apple Macintosh. Wegen der völlig anderen Hardwarebasis und des daraus resultierenden anderen Betriebssystems ist ein Vergleich zwischen diesen beiden Gruppen recht problematisch, was im Buch auch deutlich spürbar wird. Zum anderen stellt sich die Frage, warum in die Gruppe der PC-Systeme nicht auch weitere Systeme, wie SMARTWARE II, Knowledgeman oder GURU aufgenommen wurden.

Die Autoren schreiben im Vorwort auf Seite 7: „Die ... Programmpakete werden miteinander so verglichen, daß der aufmerksame Leser leicht erkennen kann, welches Programm für ihn am besten geeignet ist." Diesem Grundanliegen kann das Buch leider nicht gerecht werden, da ich die Schlußfolgerung ziehen würde, daß zumindest die vier PC-Systeme bis auf geringfügige Unterschiede einen mehr oder weniger ähnlichen Leistungsumfang haben. Das wird besonders deutlich, wenn man sich die einzelnen Vergleichstabellen ansieht. Nun ist auch das sicher eine beachtenswerte Tatsache, doch trägt sie eben kaum zur Entscheidungsfindung bei einem potentiellen Nutzer bei.

Natürlich habe ich bei der Rezension des Buches insbesondere die Verhältnisse in der DDR im Auge. Bis auf wenige Ausnahmen wird hier bisher – leider – hauptsächlich mit „adaptierter" Software gearbeitet.

Daneben werden vereinzelt auch OPEN ACCESS und Symphony genutzt. Den weitaus größeren Anwenderkreis hat dabei INSYS. Daraus ergibt sich, daß bei einem Vergleich integrierter Software besonders große Interessen bei dem System INSYS bzw. dem Vorbild SMARTWARE liegen dürften, das jedoch nicht erwähnt wird. Untersuchungen in der DDR ergaben, daß SMART sowohl FRAMEWORK als auch OPEN ACCESS und Symphony auf praktisch allen Gebieten klar überlegen ist.

Zusammenfassend möchte ich drei Punkte hervorheben: Das Buch ist methodisch hervorragend aufgebaut; die inhaltlichen Angaben sind wegen der Ähnlichkeit der Programmsysteme einerseits und der unterschiedlichen Hardwarebasis andererseits nicht allzu aussagekräftig; speziell für die DDR ist das Buch aufgrund der besonderen Hardware- und Softwarebedingungen nur von relativ geringem Interesse.

H. Kliem

Anweisungen

Einfache Anweisung	<ausdruck>;
leere Anweisung	;
Block (Verbundanweisung)	{ [<vereinbarungsliste>] [<anweisungsliste>] }
If-else-Anweisung	**if** (<ausdruck>) <anweisung> **[else** <anweisung>**]**
While-Anweisung	**while** (<ausdruck>) <anweisung>
Do-while-Anweisung	**do** <anweisung> **while** (<ausdruck>)
For-Anweisung	**for** (<ausdr>; <ausdr>; <ausdr>) <anweisung> *for (i = 0; i < N; i++) f[i] = 0;*

Switch-Anweisung	**switch** (<ausdruck>) {
	case <konstantenausdruck>: <anweisungslliste> ...
	default: <anweisungsliste>
	case <konstantenausdruck>: <anweisungsliste> ...
	}

Continue-Anweisung	**continue;**
Break-Anweisung	**break;**
Return-Anweisung	**return** [<ausdruck>];
Goto-Anweisung	**goto** <marke>; <marke>: <anweisung>

Standardfunktionen für die Tastaturein-/ausgabe

<stdio.h>

```
int getchar ( )            char *gets(s)        197
                                char *s;        200

int putchar(c)             int puts(s)          208
    int c;                      char *s;        209
```

```
int scanf(format [, pointer...])                210
    char *format;
```

Konvertierungsvorschrift

% [*] [<feldbreite>] [l|h] <konvertierungszeichen>

- % n d u o x i e f g s c [scanset]
- l = long, h = short
- maximale Feldlänge
- Wertzuweisung unterdrücken

Trennzeichen in der Formatkette werden ignoriert, alle anderen Zeichen müssen in der Eingabe vorkommen. Trennzeichen in der Eingabe wirken als Trennzeichen außer bei %c.

```
int printf(Format [, arg ...])                  206
    char *format;
```

Konvertierungsvorschrift

% [<flags>] [<feldbr> | *] [.[<genauigk> | *]] [l]<konv_z>

- long
- % d o u x X f e E g G c s
- Dezimalstellen / Anz. signif. Stellen / max. Ausgabefeldlänge
- minimale Feldbreite
- – : linksbündig
  + : Vorzeichen erzwingen
  : Platz für Vorzeichen freihalten
  # : Int-Zahlen in C-Syntax
  Punkt bei e, E, f, g, G erzwingen
  Abschließende Nullen erhalten bei g, G

Standardfunktionen für den Dateizugriff

<stdio.h>

```
FILE fopen (filename, type)                                        194
    char *filename, *type;
              "r" "r+" "a" "w" "w+" "a+"

int fclose(filepointer)                                            193
    FILE *filepointer;

int getc(filepointer)            int fgetc(filepointer)            197
    FILE *filepointer;               FILE *filepointer;            197
int putc(c, filepointer)         int fputc(c, filepointer)         208
    int c;                           int c;                        208
    FILE *filepointer;               FILE *filepointer;

int ungetc(c, filepointer)                                         220
    int c;
    FILE *filepointer;

char *fgets(s, n, filepointer)                                     200
    char *s;
    int n;
    FILE *filepointer;
int fputs(s, filepointer)                                          209
    char *s;
    FILE *filepointer;

int fread(ptr, size, nitems, filepointer)                          195
int fwrite(ptr, size, nitems, filepointer)                         195
    char *ptr;
    size_t size;                                           <types.h>
    int nitems;
    FILE *filepointer;

int fscanf(filepointer, format [, pointer....])                    210
int fprintf(filepointer, format [, arg ...])                       206
    FILE *filepointer;
    char *format;

int sscanf(s, format [, pointer ...])                              210
int sprintf(s, format [, arg ...])                                 206
    char *s, format;
```

Standardfunktionen für den Dateizugriff

<stdio.h>

```
int fseek(filepointer, offset, origin)          196
    FILE *filepointer;
    long offset;
    int origin;
long ftell(filepointer)                         196
    FILE *filepointer
void rewind(filepointer)
    FILE *filepointer                           196

int fflush(filepointer)                         193
int feof(filepointer)                           193
int ferror(filepointer)                         193
void clearerr(filepointer)                      193
int fileno(filepointer)                         193
    FILE *filepointer;
```

UNIX-Systemschnittstelle für Dateioperationen

<fcntl.h> + <types.h> + <stat.h>

```
int open(filename, oflag [, mode])              142
    char *filename;
    int oflag, mode;
        |  └ Schutzmodus (9 Bit)
        └ O_RDONLY O_WRONLY O_RDWR
int creat(filename, mode)                       142
    char *filename;
    int mode;
int close(fd)         int unlink(filename)      144
    int fd;               char *filename;       144

int read(fd, buffer, n)                         145
int write(fd, buffer, n)                        145
    int fd;
    char *buffer;
    unsigned n;

long seek(fd, offset, origin)                   145
    int fd, origin;
    long offset;
```

Zeichentyp- und Zeichenkettenfunktionen

<ctype.h>/<string.h>

```
int toupper(c)          int tolower(c)          188
int isupper(c)          int isspace(c)          190
int islower(c)          int ispunct(c)
int isalpha(c)          int iscntrl(c)
int isdigit(c)          int isprint(c)
int isxdigit(c)         int isgraph(c)
int isalnum(c)          int isascii(c)
    int c;                  int c;

char *strcat(s1, s2)    char *strncat(s1, s2, n)    215
char *strcpy(s1, s2)    char *strncpy(s1, s2, n)
int strlen(s)
int strcmp(s1, s2)      int strncmp(s1, s2, n)
char *strchr(s, c)
char *strrchr(s, c)

    char *s1, s2, s;
    int n, c;
```

Management-Funktionen

```
                                              <malloc.h>
char *malloc(size)        void free(pointer)        203
    unsigned size;            char *pointer;        203
char *calloc(nelem, elsize)                         203
    unsigned nelem, elsize;

                                              <setjmp.h>
void setjmp(env)          void longjmp(env, val)    214
    jmp_buf env;              jmp_buf env;          214
                              int val;
                                              <signal.h>
int (*signal(sig, func))()
    int sig;                                        153
    int (*func)();

                                               <stdio.h>
int system(cmd_line)
    char *cmd_line                                  217
char *getenv(name)        int putenv(string)        198
    char *name;               char *string;         209
```

Präprozessoranweisungen

Bezeichner- und Makrodefinition

```
# define <bezeichner> <zeichenfolge>
# define <makrobezeichner> (par, ...) <zeichenfolge>
# undef <bezeichner>

# define DEBUG
# define abs(x) (x < 0 ? -(x) : x)
# undef DEBUG
```

Include-Anweisung

① **# include** <<filename>>
Datei wird in Include-Verzeichnissen (Compilerflag –I) gesucht, dann im Standardverzeichnis
 # include <stdio.h>

② **# include** "<filename>"
Datei wird zuerst im gleichen Verzeichnis wie Quellfile gesucht
 # include "mydefs.h"

Anweisungen zur bedingten Compilierung

```
# ifdef <bezeichner>          # ifndef <bezeichner>
[# else]                      [# else]
# endif                       # endif
# if <konst_ausdruck> __ kann auch enthalten:
[# else]                      defined <bezeichner>
# endif                       defined (<bezeichner>)
```

Programmumgebung

```
main(argc, argv)
    int argc;
    char *argv[ ];                     /* oder: **argv; */
{
  ...
}
```

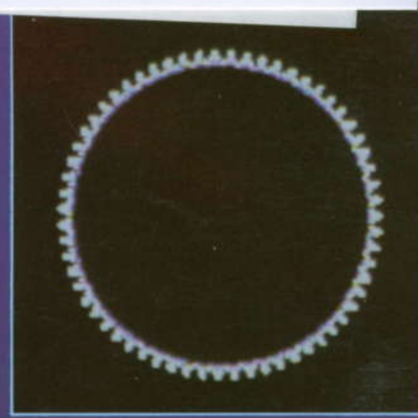
Achtung: Das Feld argv hat argc+1 Elemente, das erste zeigt auf den Programmnamen, das letzte ist NULL.

zusammengestellt von Hartmut Schäfer

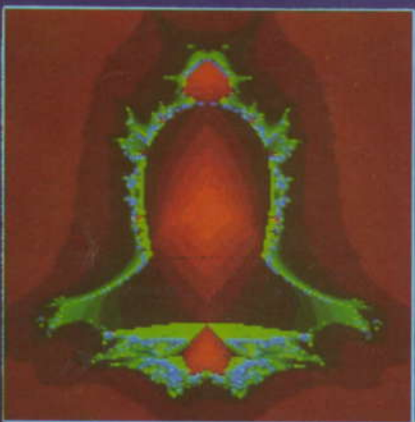
ISSN 0232-2892
Mikroprozessortechnik, Berlin 4 (1990) 5
S. 129–160

1 *Flugwanze*

2 *Einheitskreis für k = 3*

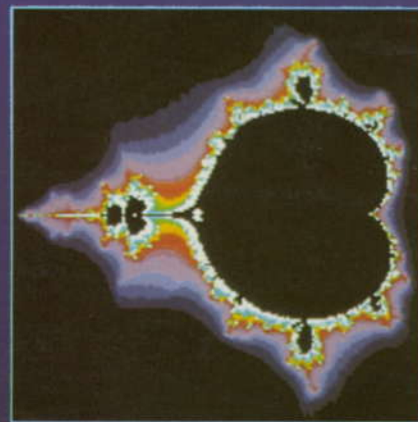


3 *Einheitskreis für k = 7*



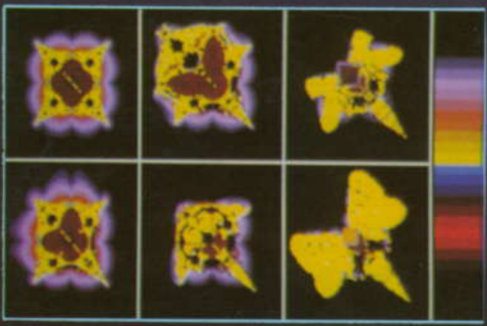
4 *Flugwanze*

5 *Gemischte Ketten aus Z-Potenzen*



6 *Zellteilung*

Lesen Sie hierzu unseren Beitrag „Fraktale als Interferenzobjekte" in diesem Heft.

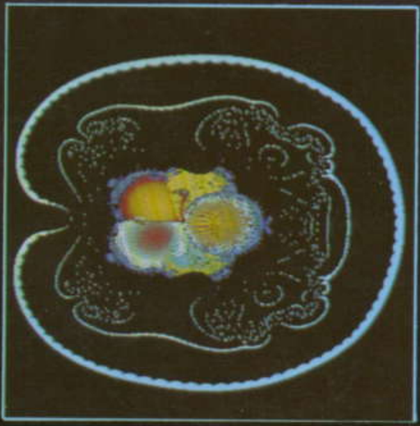


7 *Metamorphose*

8 *Schmetterling*

9 *Seestern*

10 *Landkarte Europas*



11 *Embryo*

12 *Straßen in die Unendlichkeit*

ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

MP

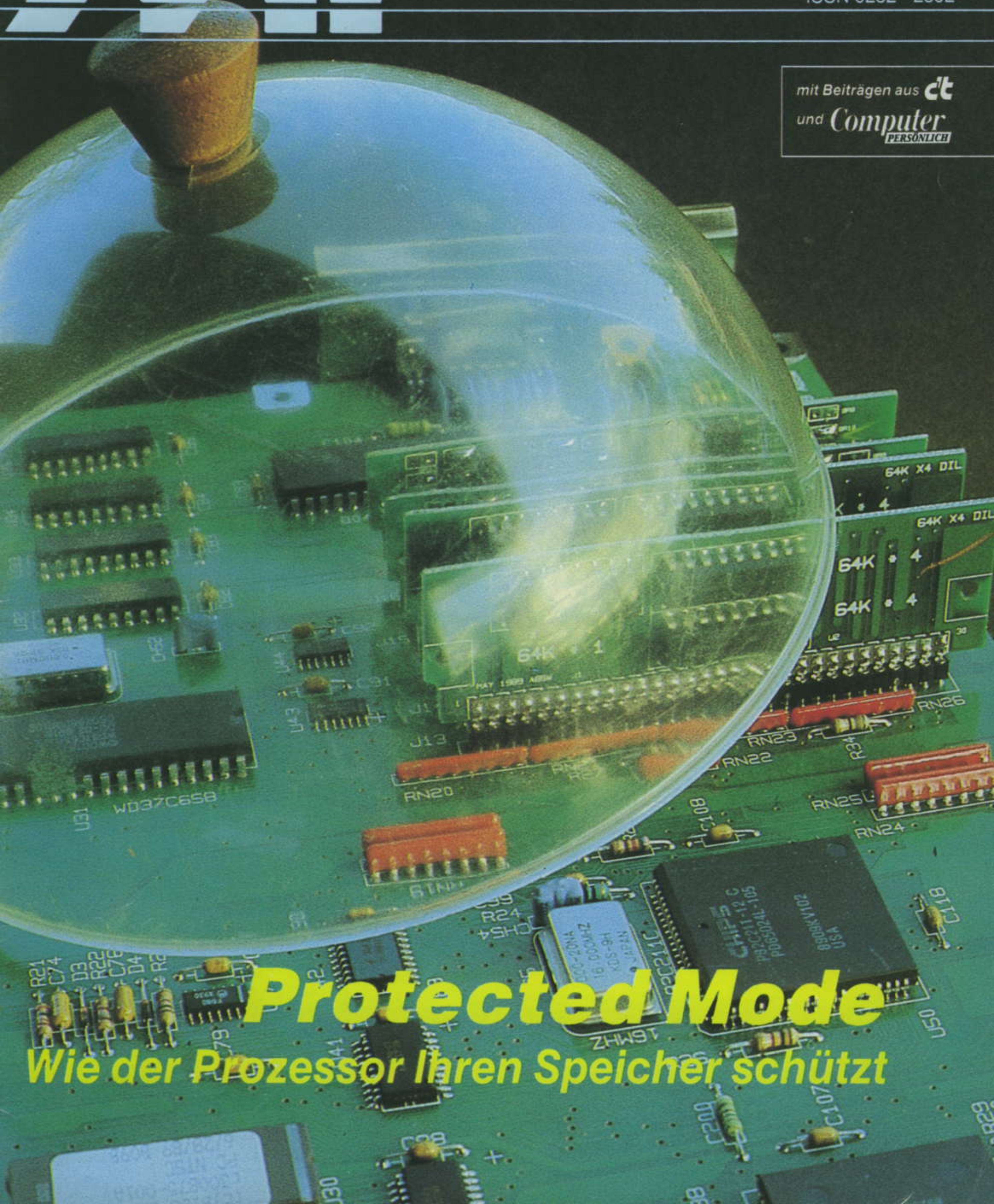
Heft 6 · 1990

Mikroprozessortechnik

VEB Verlag Technik Berlin

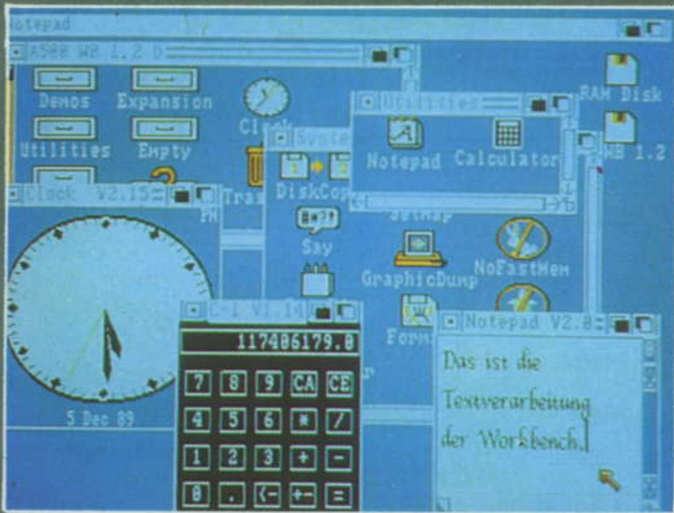
ISSN 0232-2892

mit Beiträgen aus c't und Computer PERSÖNLICH

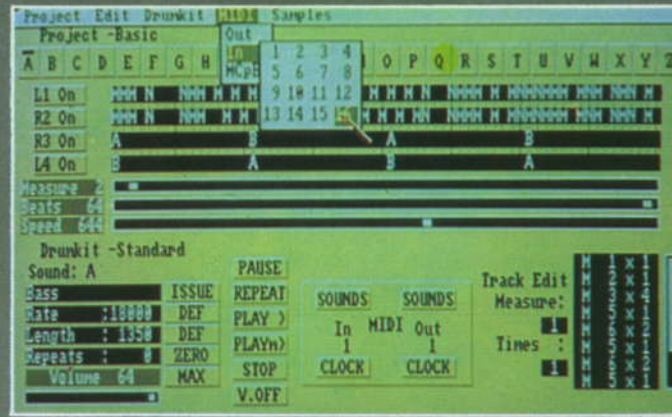


Protected Mode

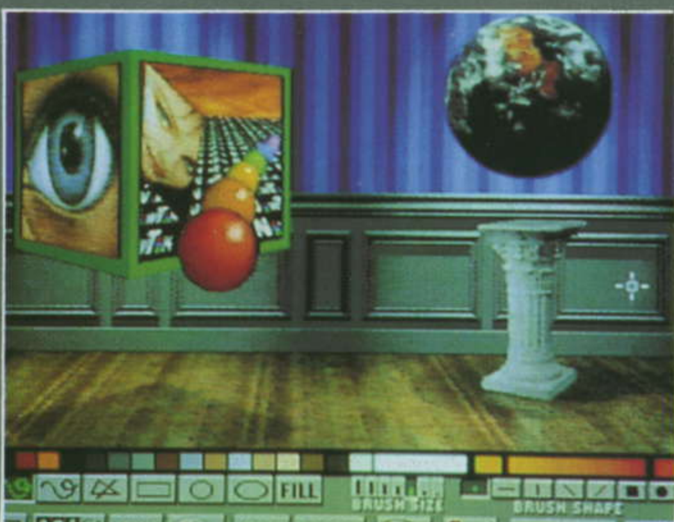
Wie der Prozessor Ihren Speicher schützt



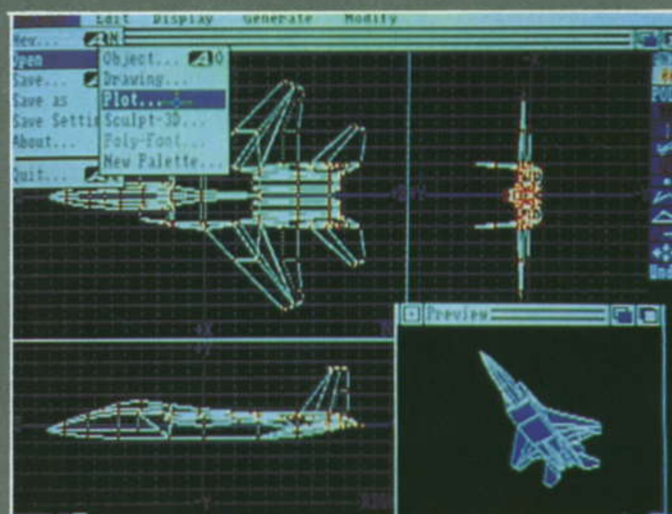
1 *Workbench – die Arbeitsoberfläche des Amiga. Mehrere Programme werden zeitgleich im Speicher gehalten.*



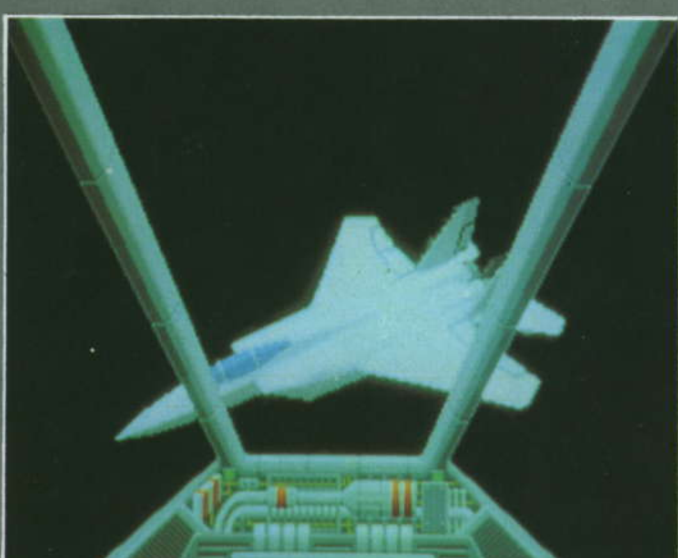
2 *ADRUM – ein Programm, das alle Funktionen einer Rhythmusmaschine erfüllt. Es wird mit gesampelten Sounds gearbeitet.*



3 *Digipaint – für das Bearbeiten digitalisierter Bilder*



4 *Modeler 3D – dient zum Erzeugen 3dimensionaler Objekte*



5 *Videospace 3D – und so sieht das Objekt in einer Animationssequenz aus.*

6 *Marbre – Grafik in 640 × 400 Punkten und 16 Graustufen*

Lesen Sie hierzu den Beitrag „Der Commodore Amiga" in diesem Heft.

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
4. Jahrgang · ISSN 0232-2892

Mikroprozessortechnik, Heft 6 · 90

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag Verlag Technik, Oranienburger Str. 13/14, DDR – 1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Hans Weiß, Verantwortlicher Redakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2870203), Hans-Joachim Hill (Tel. 2870209); Sekretariat Tel. 2870381

Gestaltung Christina Bauer

Titelfoto Hans Weiß

Lizenz-Nr. 1710 des Presse- und Informationsdienstes der Regierung der DDR

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

Redaktionsschluß 9. April 1990

AN (EDV) 49837

Erscheinungsweise monatlich 1 Heft

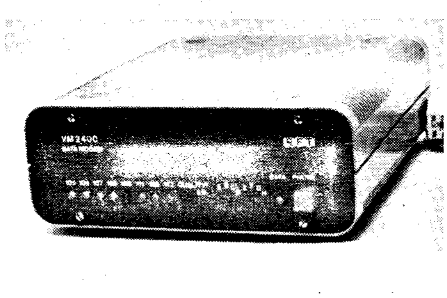
Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
DDR: sämtliche Postämter; ***SVR Albanien:*** **Direktorije** Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustřední Expedicia a Dovoz Tisku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *Republik Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Republik Ungarn:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nôi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, D-1000 Berlin; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR – 7010 Leipzig und Leipzig Book Service, Talstraße 29, DDR – 7010 Leipzig

Zum Titelbild

Wie der Prozessor Ihren Speicher schützt

Der Prozessor U 80601 vom Kombinat Mikroelektronik Erfurt arbeitet im EC 1835 gewöhnlich im Normal-Modus (Real Mode). Aber beispielsweise für die Ansteuerung von RAMs oberhalb des 640-KByte-Bereiches – als Extended Memory bezeichnet – schaltet dieser Prozessor in den geschützten Modus (Protected Mode) um. Hier werden jeder auszuführenden (Teil-)Aufgabe getrennte, gegeneinander geschützte Speicherbereiche zugeordnet. Deshalb ist dieser Modus auch eine Voraussetzung für den Multitask-Betrieb. Der Beitrag auf der **Seite 163** demonstriert die Funktion des Prozessors 80601 im Protected Mode an Hand von einfachen Beispielen, die mit der MS-DOS-Entwicklungstechnik für Assemblerprogramme entwickelt worden sind.



International längst üblich: Mailboxen statt Eintippen von Hexdumps, Online-Datenbanken statt Bücherwälzen, elektronische Postwurfsendungen statt blendfreier Kontaktarmut. Der Beitrag „Datenübertragung leicht gemacht?" auf der **Seite 192** befaßt sich mit den rechtlichen und technischen Voraussetzungen für den Einsatz von Modems.

Vorschau

Für MP 7/1990 bereiten wir für Sie unter anderem Beiträge zu folgenden Themen vor:

- Debugger
- Topspeed-Modula-2
- Centronics-Schnittstellenumschalter
- Neuer Kurs: Lisp
- Leserumfrage mit Verlosung eines Personalcomputers

Inhalt

MP-Info 162

Günter Jorke, Siegfried Melzig:
Der Protected Mode 163

Mit 66 MIPS den Echtzeitproblemen zu Leibe ... 168

Günter Sedlacek, Hartmut Lubrich:
Der Gate-Array-Schaltkreis U 5303 FC303 171

Thomas Bölkow:
Der Commodore Amiga 173

MP-Kurs:
Wolfram Schulze, Uwe Schulze:
Einführung in MS-DOS (Teil 2) 179

Horst Beidatsch, Dieter Simon:
Schnittstellen von Turbo-C zu MS-DOS 179

Wolfgang Börner:
Alles zur gleichen Zeit 181

Wegbereiter der Informatik
Friedrich Adolf Willers 183

Bernhard Brandt:
Inline eingeschränkt 184

Neue Electronic-Publishing-Werkzeuge 184

MP-Computer-Club 185
Bernd Matzke:
Einzelpunktgrafik für den A 7100
Christian Schiewe:
KC 85/1 generiert 24 × 80 Zeichen
Thomas Johannsen:
Sicheres Entriegeln von Diskettenlaufwerken
Uwe Hampel:
Tastaturtreiber-Patch
Jens Jacobi:
Fehler in Turbo-Pascal 5.0

MP-Börse 188

Entwicklungen und Tendenzen 190

Datenübertragung leicht gemacht? 192

Liebe Leserinnen
und Leser!

Aufgrund zahlreicher Anfragen an die Redaktion möchten wir Sie an dieser Stelle darüber informieren, daß es keine Absichten gibt, den gegenwärtigen Preis der Mikroprozessortechnik zu verändern.
Gegenüber vielen langjährigen DDR-Zeitschriften „durfte" für die MP als neues Produkt 1987 ein realistischer Preis bestimmt werden. Auf dieser Grundlage wollen wir die neuen Möglichkeiten nutzen, die Zeitschrift für Sie schrittweise noch attraktiver zu machen – beispielsweise mit mehr Farbe ab dem nächsten Heft und zusätzlichen Seiten für Anzeigen.
Für das Juli-Heft haben wir eine Leserbefragung vorgesehen, um Ihre Wünsche gründlicher kennenzulernen. Übrigens können Sie dabei einen Personalcomputer AT gewinnen – lassen Sie sich diese Chance nicht entgehen. *Ihre Redaktion MP*

SYSTEC 90: Computerintegration in der Industrie

Nachdem bereits vor zwei Jahren die zweite internationale Fachmesse für Computerintegration im Unternehmen hohe Erfolgszahlen aufweisen konnte, werden auch zur diesjährigen Veranstaltung vom 22. bis 26. Oktober 1990 wieder außerordentlich gute Ergebnisse erwartet. Bei einem Flächenwachstum von 20 Prozent stehen diesmal die Hallen 1–21 (1988: 1–16) zur Verfügung. An fünf Messetagen (1988: 4 Messetage) werden wieder alle namhaften Hersteller ihre Produkte und Dienstleistungen präsentieren.
Wenn 1988 89,9 Prozent der Aussteller der SYSTEC ein positives Ergebnis bescheinigten, 97 Prozent von ihnen neue Kunden erhielten und 62 Prozent von konkreten Exportkontakten berichteten, dann ist davon auszugehen, daß die SYSTEC, als erste CIM-Messe Europas, Ausstellern ebenso wie Fachbesuchern auch im Jahre 1990 wieder jene Erfolge bescheren wird, die sie verspricht. Mit der Zielsetzung, computergestützte Fertigungstechnologien überschaubar zu machen, werden die zugehörigen Einzeldisziplinen wie CAD, CAM, PPS und ihre unterschiedlichen Querverbindungen in allen Ausprägungen dargestellt.
Für die SYSTEC 90 erwartet die Münchener Messe- und Ausstellungsgesellschaft 550 Aussteller und 35 000 Besucher. *MP*

Lizensierung von Wordstar-Kopien

Die Firma Wordstar International GmbH (ehem. Micropro) bietet ab sofort die Möglichkeit der Legalisierung aller in der DDR genutzten illegalen Kopien ihrer Produkte. Die Lizenznahme erfolgt über die Firma Electronic Chemnitz, autorisierter Distributor für das Gebiet der DDR. Die genauen Konditionen (wie Update-Recht und Aufnahme in das Wordstar-Informationssystem) werden auf Anfrage gern von der Firma Electronic Chemnitz mitgeteilt bzw. sind aus den einschlägigen Veröffentlichungen zu entnehmen.
Die Möglichkeit der Lizensierung der illegalen Kopien gilt vorläufig nur bis zum 30. Juni 1990. Das Update-Recht auf die Version 5.5. besteht bis zum 15. August 1990.
Wordstar International GmbH versichert, daß aus der Lizensierung weder gerichtliche noch privatrechtliche Folgen resultieren.

Kontakt: Electronic Chemnitz, PSF 649, Karl-Marx-Stadt, 9010; Tel. 5 61 27 70 *Wötzel*

Digital Equipment eröffnet Märkte

Die Digital Equipment Corporation (DEC) hat zusammen mit dem KFKI, dem zentralen Forschungsinstitut für Physik der ungarischen Akademie der Wissenschaften, und dem DV-Unternehmen Szamalk ein Joint-Venture-Unternehmen zum Vertrieb von Computersystemen und DV-Lösungen in Ungarn gegründet. Das neue Unternehmen mit Namen Digital Equipment (Magyarorszag) Kft. wird zunächst die erfolgreiche Computerreihe MicroVAX vermarkten. Neben dem Vertrieb dieser Produkte wird das Joint-Venture-Unternehmen eine Reihe von Dienstleistungen wie Hardware-Wartung, Schulung und Software-Unterstützung anbieten. 51 Prozent der Unternehmensanteile liegen in den Händen von Digital. KFKI und Szamalk sind zu jeweils 24,5 Prozent beteiligt. Digital hat darüber hinaus die Option, das Joint Venture ganz zu übernehmen und zu einer hundertprozentigen Tochtergesellschaft zu machen.
Mit der Einrichtung einer DDR-Geschäftsstelle zum Auftakt der CeBIT '90 unterstreicht Digital Equipment zusätzlich die Bedeutung der sich öffnenden Ostmärkte für ihre langfristige Unternehmensstrategie. Von Westberlin aus wird die neue Geschäftsstelle von DEC das Unternehmen auf den DDR-Markt vorbereiten und seinen Vertriebspartnern Beratungs- und Unterstützungsleistungen bieten. Noch im laufenden Geschäftsjahr plant Digital die Eröffnung eines Büros in Dresden für den Raum Dresden/Leipzig. *MP*

Wordcraft in der DDR

Für das von einem englischen Hersteller entwickelte Textverarbeitungsprogramm Wordcraft bietet die Firma JOLO DATA einschließlich des eigenen DTP-Zusatzes Imagemaster für die DDR folgende Regelung an:
1. In der DDR darf das komplette Programm der Version 3 (5 Disketten) der Textverarbeitung Wordcraft frei kopiert werden.
2. Die Anwender dieser Textverarbeitung werden offiziell registriert und erhalten somit die volle Software-Unterstützung vom DDR-Vertrieb.
3. Die Anwender haben zusätzlich die Möglichkeit, bei umfangreicheren Arbeiten ein Handbuch oder auch zusätzliche Druckertreiber zu Selbstkosten zu beziehen.
4. Weiterentwickelte Versionen oder interessante Programmzusätze können die Wordcraft-Anwender gegen einen »Umsteiger-Preis« offiziell erwerben.

Die Informationsunterlagen können gegen Einsendung von ausreichendem Rückporto von Rüdiger Trenkler, Lotzestraße 18 in 8600 Bautzen angefordert werden. *MP*

ORGATEC Köln 1990: fast ausgebucht

Ein wachsender Bedarf für die Neueinrichtung von Büros und dementsprechend günstige konjunkturelle Rahmenbedingungen in den meisten europäischen Ländern kennzeichnen die Vorbereitungen zur ORGATEC, Internationale Büromesse Köln, vom 25. bis 30. Oktober 1990. Mit rund 2 000 ausstellenden Unternehmen aus mehr als 30 Ländern und einer um 9 Prozent auf 250 000 m² erweiterten Ausstellungsfläche ist die Messe bereits jetzt fast komplett ausgebucht.
Als führende internationale Fachmesse für den Gesamtmarkt Büro kommt der ORGATEC eine auch in Zukunft weiter wachsende Bedeutung zu. Rund 60 Prozent aller Büroarbeitsplätze entsprechen noch nicht den heutigen Anforderungen an Technik und Einrichtung. Der Bedarf für die Neueinrichtung wird bis Ende dieses Jahrzehnts auf etwa 40 Millionen geschätzt. Das „menschliche" Büro steht dabei im Mittelpunkt des Interesses.

Kontakt: Messe- und Ausstellungs-Ges. m. b. H. Köln, Messeplatz 1, Postfach 21 07 60, D-5000 Köln 21, Tel. 821-0. *MP*

Studie über Datenverarbeitungslandschaft der DDR

Die UniWare Computer GmbH bietet eine Studie über die DV-Landschaft der DDR an, die das Verständnis für das Arbeitsumfeld und die Ausbildung der Mitarbeiter in dieser Branche vermittelt. Die Studie ist ein Arbeitsergebnis der Kooperation zwischen der UniWare Computer GmbH, Berlin-West, und der GKI-Gesellschaft für offene Kommunikations- und Informationssysteme mbH, Berlin-Ost.
Die Studie kann ab sofort bei UniWare, Kantstraße 152, D-1000 Berlin 12 für DM 1350,– + MwSt. bezogen werden. Sie gibt einen Überblick über die Rechentechnik-Eigenproduktion, deren Entwicklungsstand und Leistungsfähigkeit, über die Entwicklung und Verbreitung von UNIX, beschreibt die Situation der Computerkommunikation, stellt die vorhandene Datenbank-Software vor und geht schließlich auf die relevanten Berufsbilder und Ausbildungsgänge in der DDR ein. Nach der Beschreibung der Tätigkeitsfelder der Kombinate Datenverarbeitung und Robotron bilden die wichtigen Adressen der Datenverarbeitungsbetriebe sowie der wissenschaftlichen Einrichtungen den Abschluß. *MP*

NETWORK-Jahresprogramme

Von der NETWORK GmbH, Veranstalter von Fachausstellungen und Tagungen, werden 1990/91 unter anderem folgende Veranstaltungen vorbereitet:

UNIX in Deutschland
mit GUUG-Jahrestagung '90
Rhein-Main-Hallen Wiesbaden
4. bis 6. September 1990

INTERDECK '91
Messe für VAX- und PDP11-Anwender zusammen mit dem DECUS-Symposium Hannover
23. bis 25. April 1991

UNIX in Deutschland
mit GUUG-Jahrestagung '91
Rhein-Main-Hallen Wiesbaden
24. bis 26. September 1991

Weitere Informationen über: NETWORK GmbH, Wilhelm-Suhr-Straße 14, D-3055 Hagenburg, Tel. 70 57, Fax (05033) 7944, Telex 92 45 45 *MP*

c't und MP auf der LFM '90

Eine große Zahl von Lesern und Interessierten nutzte die Gelegenheit des Lesertreffs, den wir in MP 2/90 angekündigt hatten, um sich in Gesprächen mit Redakteuren und anhand von Probeheften tiefergehend über die beiden Publikationen zu informieren. *Foto: Hemke*

Programmierung des Prozessors 80601

Der Protected Mode

Dr. Günter Jorke, Siegfried Melzig, Rostock

Der Aufbau und die Funktion des 16-Bit-Prozessors U 80601 sind in der MP in /1/ und /2/ vorgestellt worden. Im folgenden werden Programmiererfahrungen mit der Arbeit dieses Prozessors im Protected Mode (geschützter Modus) mitgeteilt. Im Protected Mode werden die Speicherbereiche mehrerer Tasks (Aufgaben) gegeneinander geschützt. Der Betrieb des Prozessors im Protected Mode ist deshalb die Voraussetzung für echtes Multitasking (gleichzeitiger Ablauf mehrerer Programme).

Versuch macht klug

Die Assemblerprogrammentwicklung im Protected Mode wird im Normalfall durch spezielle Programmentwicklungswerkzeuge unterstützt. Für einfache Programme ist jedoch auch eine Assemblerprogrammentwicklung mit einfachen Entwicklungswerkzeugen des Betriebssystems MS-DOS möglich. Dieses Vorgehen besitzt den Vorteil, daß man alle Funktionen des Prozessors bis hin zum Befehlsniveau kennenlernt.
Der Protected Mode kann durch das Setzen des Bits 0 (PE, protected mode enable) in einem *Maschinenstatuswortregister* eingestellt werden. Dazu dienen zum Beispiel die Befehle

```
SMSW AX
OR   AX,1
LMSW AX
```

mit denen das Maschinenstatuswort in das Register AX geladen, das Bit 0 mit überschrieben und in das Maschinenstatusregister zurückgeschrieben wird.
Bevor wir den Protected Mode einstellen können, müssen einige Datenstrukturen im Speicher angelegt werden.

Metamorphose der Segmentregister

Im Protected Mode ist der direkt ansprechbare Speicheradreßraum auf 16 MByte erhöht worden. Der Speicher wird mit einer Vielzahl von Segmenten verwaltet, die maximal 64 KByte groß sein können.
Für den Anwender wird diese Umstellung in einem Funktionswandel der vom Prozessor 8086 bekannten Segmentregister in *Selektorregister* ersichtlich. Jedes Selektorregister CS, ES, DS und SS kann in den oberen 13 Bitstellen einen Index auf zwei Deskriptortabellen GDT oder LDT enthalten. Die globale Deskriptortabelle GDT und die lokale Deskriptortabelle LDT enthalten im Speicher bis zu 8192 Segmentdeskriptoren. Ein 8 Byte großer Segmentdeskriptor enthält nach Bild 1 eine 24-Bit-Basisadresse, die Segmentgröße und weitere Informationen des Segmentes. In einem Steuerbyte (access byte) sind der Typ und die Zugriffsinformationen des Segmentes eingetragen.

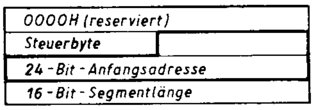


Bild 1 Segmentdeskriptor

Die physische 24-Bit-Zugriffsadresse auf den Speicher wird aus der Offsetadresse und der über den Selektor im Deskriptor ausgewählten Basisadresse gebildet. Dabei gelten die in der 8086-Assemblerprogrammierung üblichen Speicheradressierungsformen.
Wenn nun aber bei jedem Speicherzugriff die Zugriffsadresse über im Speicher abgelegte Tabellen gebildet werden müßte, würde die Adreßraumvergrößerung mit einer erhöhten Zugriffszeit bezahlt werden. Deshalb werden die 6 signifikanten Bytes der Segmentdeskriptoren für die 4 Selektorregister im Prozessor abgespeichert. Zu diesem Zweck enthält der 80601 für jedes Selektorregister einen 6 Byte großen Cacheanteil („verstecktes Selektorregister") nach Bild 2. Die Abarbeitung von Befehlen, die den Inhalt eines Selektorregisters ändern (z. B. MOV ES,AX; POP DS; JMP FAR), schließt nun das Kopieren des durch den neuen Selektor ausgewählten Segmentdeskriptors in den Cacheteil des Selektorregisters ein.

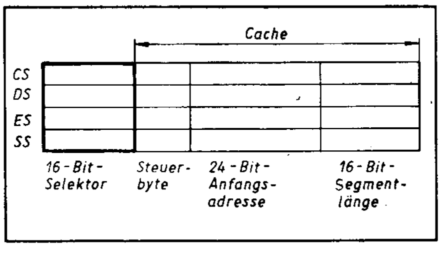


Bild 2 Selektorregister

Das Bit 2 im Selektorregister entscheidet, ob der Selektor auf die globale (Bit 2 = 0) oder die lokale Deskriptortabelle zugreift. Anfangsadresse und Länge der GDT werden im Prozessor in dem 5-Byte-Register GDTR abgespeichert.

Geheimniskrämerei im Speicher

Die neben der Basisadresse im Prozessor verfügbaren Informationen über ein Segment sind die Voraussetzung für eine hardwaremäßige Zugriffskontrolle auf den Speicher. Die Längeninformation des Segmentes wird genutzt, um die Offsetadresse auf einen oberen Grenzwert zu überprüfen. Mit Hilfe des Steuerbytes werden Segmenttypen definiert. Zunächst betrachten wir die Segmenttypen Codesegment und Datensegment.
Der Prozessor kann nur Programme in *Codesegmenten* (executable segments) abarbeiten. Nach Bild 3 können Codesegmente so definiert werden, daß sie nicht als Daten lesbar sind (EO, executable only). Das Bit A markiert einen ausgeführten Zugriff auf das Segment.
Mit dem Bit P wird ein virtueller, logischer Speicherraum organisiert, der wesentlich größer als der physische Speicher ist. Über die beiden Deskriptortabellen GDT und LDT kann der Prozessor zu einer bestimmten Zeit auf 2 × 8192 Segmente von je maximal 64 KByte logisch zugreifen. Daraus resultiert die Angabe von 1 GByte virtueller Speicher (pro Task). Bei einem Zugriff auf ein Segment, das nicht im physischen Speicher steht, liefert der Prozessor Anschlußmöglichkeiten für Routinen, die eine Speicherladeoperation für das Segment einschieben. Die Diskussion der weiteren Eintragungen des Steuerbytes verschieben wir auf die Besprechung der Schutzmechanismen.
In einem Datensegmentdeskriptor nach Bild 4 kann der Schreibzugriff auf ein Datensegment verboten werden (RO, read only). Außerdem werden Stapeldatensegmente von einfachen Datensegmenten mit dem Wert des Bits ED unterschieden. Beim Zugriff auf ein Stapelsegment mit ED = 1 wird kontrolliert, ob die Offsetadresse unterhalb der als „Längeninformation" im Deskriptor angegebenen Untergrenze liegt. In diesem Fall wird eine Ausnahmebehandlung eingeleitet.
Besteht beispielsweise der Wunsch, ein EO-Segment zu lesen, kann mit einem zusätzlichen Alias-Deskriptor für dieses Segment diese Zugriffsrechte definieren.

Erster Versuch

Die folgenden Testbeispiele sind auf einem Single-Board-Computer 80601 abgearbeitet worden. Dieser verwendet über ein Parallelinterface einen A 5120 als Terminal.
Der EPROM-residente Monitor des SBC-80601 enthält die Möglichkeit, unter MS-DOS erzeugte Exe-Dateien zu laden.

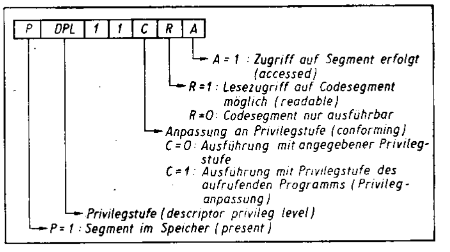


Bild 3 Steuerbyte für Codesegmentdeskriptor

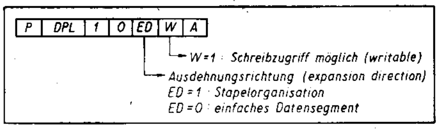


Bild 4 Steuerbyte für Datensegmentdeskriptor

Die Programmentwicklung erfolgte unter dem Betriebssystem MS-DOS mit dem Assembler MASM V 5.0. Befehle, die die Selektoren auf Segmentdeskriptoren enthalten, müssen dabei im Maschinencode in Form von DB-Anweisungen angegeben werden.
Das erste Programmbeispiel enthält den Übergang in den Protected Mode und zeigt die Anwendung der Selektoren beim Zugriff auf die Datensegmente und beim Unterprogrammaufruf innerhalb eines Codesegmentes. Die benutzten Selektoren beziehen sich vorerst nur auf die globale Deskriptortabelle (GDT). In Bild 5 sind die zur Zeit der Programmerarbeitung möglichen Einträge in die GDT angegeben. Der erste Eintrag ist der Null-Deskriptor. Der zweite Eintrag ist obligatorisch der Deskriptor der GDT. Um die GDT später erweitern zu können (Schreibzugriff), deklarieren wir den Speicherbereich der GDT als Datensegment und vermerken im Steuerbyte:

Read/Write, Privileg Level 0.

```
PAGE
 ;Global Descriptor Table

seg_gdt            SEGMENT PARA

;GDT(0)            Null-Deskriptor
        db         0,0
        db         0,0
        db         0,0
        dw         0          ;reserviert fuer 80386
;GDT(1)            Verweis auf sich selbst
t_gdt   dw         300H       ;Laenge
b_gdt   dw         0          ;Basisadresse Bit  0-15
        db         0          ;Basisadresse Bit 16-23
        db         92H        ;A: 10010010B
        dw         0
;GDT(2)            IDT-Beschreiber
t_idt   dw         1000H      ;Laenge
b_idt   dw         0          ;Basisadresse Bit  0-15
        db         0          ;Basisadresse Bit 16-23
        db         92H        ;A: 10010010B
        dw         0
;GDT(3)            Stacksegment PL 0
        dw         200H       ;"Boden" des Stack
b_stack dw         0          ;Basisadresse Bit  0-15
        db         0          ;Basisadresse Bit 16-23
        db         096H       ;A: 10010110B
        dw         0
;GDT(4)            Datensegment Hauptprogramm
        dw         100H       ;Laenge
b_data  dw         0          ;Basisadresse Bit  0-15
        db         0          ;Basisadresse Bit 16-23
        db         092H       ;A: 10100010B
        dw         0
;GDT(5)            Codesegment Hauptprogramm
        dw         100H       ;Laenge
b_code  dw         0          ;Basisadresse Bit  0-15
        db         0          ;Basisadresse Bit 16-23
        db         09EH       ;A: 1001110B
        dw         0
;GDT(6)            Codesegment Ausgabe
        dw         100H       ;Laenge
b_out   dw         0          ;Basisadresse Bit  0-15
        db         0          ;Basisadresse Bit 16-23
        db         09AH       ;A: 1001110B
        dw         0
```

Bild 5 Aufbau der globalen Deskriptortabelle

```
PAGE 66,80
;------------------------------------
;        Initialisierung 80286 fuer
;     Protected Virtuell Address Mode
;------------------------------------
.286p

;Selektoren GDT

sel_stack       equu      18H      ;GDT(3)
sel_data        equ       20H      ;GDT(4)
sel_code        equ       28H      ;GDT(5)
sel_out         equ       30H      ;GDT(6)
sel_c_exc       equ       60H      ;GDT(12)

;Hauptprogramm - Stack

seg_stack       SEGMENT STACK

 db      256 dup('S')

seg_stack       ENDS

;Hauptprogramm - Codesegment
;Umschalten in Protected Mode

seg_code        SEGMENT 'init'
                ASSUME CS:seg_code, DS:seg_gdt
start:  cli
;Eintrag der Segmentbasisadressen in die GDT
;--------------------------------------------
        mov     cl,4
        mov     ax,seg_gdt
        mov     ds,ax
        shl     ax,cl
        mov     word ptr b_gdt,ax
        mov     ax,seg_code
        shl     ax,cl
        mov     word ptr b_code,ax
        mov     ax,seg_idt
        shl     ax,cl
        mov     word ptr b_idt,ax
        mov     ax,seg_stack
        shl     ax,cl
        mov     word ptr b_stack,ax
        mov     ax,seg_data
        shl     ax,cl
        mov     word ptr b_data,ax
        mov     ax,seg_out
        shl     ax,cl
        mov     word ptr b_out,ax
;Basisadr. und Laenge fuer GDTR, IDTR aus GDT
        mov     ax,seg_gdt
        mov     ds,ax
;GDTR, IDTR laden
        lgdt    qword ptr t_gdt
        lidt    qword ptr t_idt
;Umschalten in Protected Mode
        smsw    ax
        or      ax,1
        lmsw    ax
```

Bild 6 80601-Programm mit der Umschaltung aus dem Real Address Mode in den Protected Mode ▶

Der dritte Eintrag (GDT (2)) beschreibt die Interruptdeskriptortabelle (s. u.).
Im Datensegmentdeskriptor (GDT (4)) legen wir die Länge mit 100H fest. Das Datensegment liegt im Speicher somit von Segmentbasisadresse +0 bis Segmentbasisadresse +100H. Der Versuch, diese Grenze nach oben oder unten zu überschreiten, führt zu einer Ausnahmebehandlung (s. u.). Im Stacksegmentdeskriptor (GDT (3)) tragen wir für die „Länge" 80H ein. Das ist die untere Segmentgrenze unseres Stacksegmentes – der „Boden" des Stacks. Die obere Segmentgrenze jedes Stacksegmentes ist FFFFH. Das Stacksegment reicht von der Offsetadresse 80H bis zur Offsetadresse FFFFH.
Mit Eintrag GDT (5) wird das Codesegment des Hauptprogramms und mit GDT (6) das der Ausgaberoutine beschrieben. Das Steuerbyte spezifiziert bei beiden:

Codesegment, Executable/Read, Privileg Level 0

Die Ausführung des Programms nach Bild 6 beginnt im automatisch eingestellten Real Mode. Da die Segmentadressen erst nach der dynamischen Speicherzuweisung vor der Programmausführung feststehen, können die Segmentbasisadressen in die Deskriptoren der GDT erst jetzt eingetragen werden.
Spätestens zum Zeitpunkt des ersten Modifierens eines der Selektorregister müssen Basisadresse und Länge der GDT im Prozessor im Register GDTR eingetragen sein. Mit dem Befehl LGDT wird das GDT-Register geladen. Der Befehl LIDT überträgt Basisadresse und Länge der Interruptdeskriptortabelle in das Register IDTR. Danach wird der Protected Mode eingestellt.
Im Programm nach Bild 7 kann im Protected Mode zunächst mit den schon im Real Mode initialisierten Segmenten weitergearbeitet werden. Jeder nachfolgende Segmentwechsel (z. B. mit MOV DS,AX) bezieht sich jedoch auf den selektierten Deskriptor und unterliegt der Prüfung auf Zugriffsrecht, Schutzrechte und Präsenz im Hauptspeicher.
Zunächst initialisieren wir Stack- und Datenselektor. Der Selektor in DS zeigt jetzt auf einen Segmentdeskriptor in der GDT, der das Datensegment seg_data beschreibt. Vor dem ersten Zugriff auf dieses Segment teilen wir das dem Assembler mit der ASSUME-Anweisung mit. Den Stackpointer laden wir mit 100H und nutzen das Stacksegment von Basisadresse + 80H bis Basisadresse + 100H. Die obere Grenze unseres Stacks, die wir so festgelegt haben, wird damit nicht vom Prozessor überwacht.
Der Aufruf der Ausgaberoutine mit

CALL FAR Selektor: offset

wird analog zum CALL FAR im Real Mode ausgeführt. Codeselektor und Instructionpointer der aufrufenden Routine werden auf dem Stack abgelegt und Codeselektor und Instructionpointer der aufgerufenen Routine in die Register CS und IP geladen. Dabei wird der Deskriptor des Zielcodesegmentes aus der GDT in den Cacheteil des Selektorregisters CS übernommen. Bei Rückkehr aus dieser Routine mit RET FAR werden IP und der Codesegmentselektor vom Stack geholt und geladen. Dabei wird der Deskriptor aus der GDT wieder in den Cacheteil von CS zurückgeschrieben.
Auch hier erfolgt wieder eine Prüfung auf Zugriffsrecht usw. anhand des Segmentdeskriptoreintrages, auf den der Selektor zeigt. Wir müssen also nach dem Übergang in den Protected Mode dafür sorgen, daß vor dem Aufruf der Ausgaberoutine der gültige Selektor unseres Hauptprogramms im Register CS steht (Bild 7). Dafür werden der aktuelle Codesegmentselektor und der Instructionspointer in den Stack eingetragen und durch einen RETF-Befehl in die Register geladen.

```
PAGE
;--------------------------------
;umgeschaltet in Protected Mode
;--------------------------------
;Selektoren fuer Stack- und Datensegment laden
        mov     ax,sel_stack
        mov     ss,ax
        mov     sp,100H
        mov     ax,sel_data
        mov     ds,ax
        ASSUME DS:seg_data
;Selektor des aktuellen Codesegmentes laden
        mov     ax,sel_code
        push    ax
        mov     ax,offset main
        push    ax
        retf
;Ausgabe einer Zeichenkette auf Bildschirm
main:   mov     si,offset string
main1:  mov     al,byte ptr[si]
;       CALLF   CONOUT
        db      9AH
        dw      offset conout
        dw      sel_out
        inc     si
        cmp     al,0
        jnz     main1
        jmp     main2

seg_code        ENDS

;Hauptprogramm - Datensegment

seg_data        SEGMENT PARA

string  db      0CH,0AH,0DH
        db      'Protected Mode',0

seg_data        ENDS

;Byteausgabe auf Bildschirm

seg_out         SEGMENT
                ASSUME CS:seg_out
conout:
.list
        retf
seg_out         ENDS
```

Bild 7 80601-Programm im Protected Mode

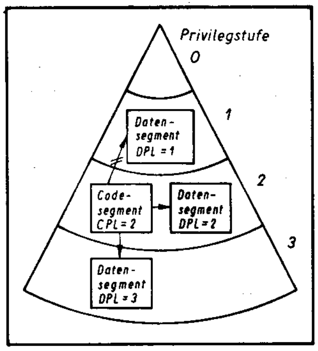


Bild 8 Privilegregeln für den Datensegmentzugriff

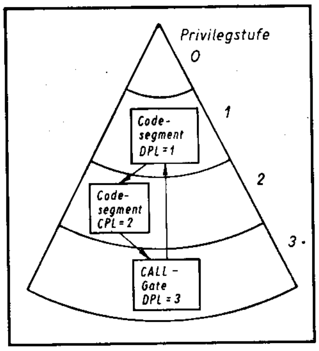


Bild 10 Privilegregel für Codesegmentansprung über CALL-Gates

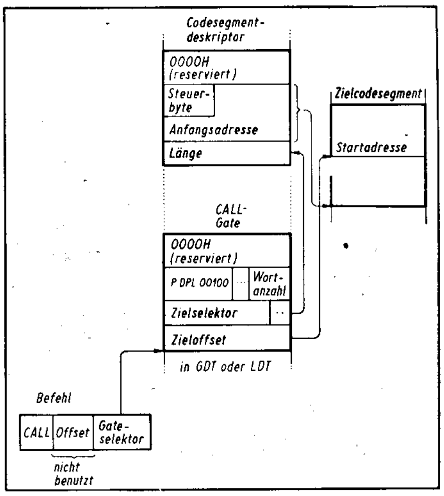


Bild 9 Intersegmentsprung über CALL-Gates

Speicher auf Zuteilung

Eine weitere wesentliche Funktion der Speicherverwaltung bezieht sich auf die Verteilung des Speichers auf verschiedene Aufgaben. Einer Aufgabe werden die Speicherressourcen durch die Bildung einer *Task* zugewiesen. Hierbei umfaßt der Begriff Task etwa die Eigenschaften der Objekttypen Job und Task eines Echtzeitbetriebssystems, wie RMX86 oder BOS 1810.
Jede Task erhält eine eigene lokale Deskriptortabelle. Die Programme innerhalb einer Task können deshalb nur auf globale Segmente, die in der GDT beschrieben sind, oder auf ihre eigenen lokalen Segmente über die LDT zugreifen.
Innerhalb jeder Task besitzt jedes Segment eine der vier Privilegstufen 0...3.
Die Privilegstufe eines Segments wird in den unteren zwei Bits des Segmentselektors in der Deskriptortabelle als Deskriptorprivilegniveau (DPL) angegeben.
Andererseits ist jede Programmausführung durch die Privilegstufe des dazugehörigen Codesegmentes als Current Privileg Level (CPL) gekennzeichnet. CPL wird durch die beiden unteren Bits des CS-Selektorregisters festgelegt.
Über den Privilegmechanismus wird der Datenzugriff weiter eingeschränkt. Ein Programm darf nur auf Datensegmente zugreifen, die die gleiche oder eine niedrigere Privilegstufe (wertmäßig größer) besitzen als das Current Privileg Level:

$$CPL \leqq DPL_{Datensegment}$$

Bild 8 zeigt mögliche und zulässige Zugriffe auf Datensegmente.
Sprünge und Unterprogrammsprünge auf andere Codesegmente sind mit den einfachen FAR-Sprüngen, bei denen der Selektor des Zielcodesegmentes angegeben wird, nur in der gleichen Privilegstufe möglich.

Wie man große Sprünge macht

Ein Unterprogrammsprung in eine höhere Privilegstufe kann nur mit *Call-Gates* ausgeführt werden. Dabei kehrt das Programm auf die Privilegstufe des aufrufenden Programms zurück. Damit kann ein Anwenderprogramm auf einer niedrigen Privilegstufe, zum Beispiel 3, auf Bibliotheken des Betriebssystems auf höheren Privilegstufen zugreifen.
Die Privilegregeln unter Benutzung von CALL-Gates lautet:

$$CPL \geqq DPL_{Zielcodesegment}$$

Ein CALL-Gate in der GDT oder der LDT ist nach Bild 9 einem Segmentdeskriptor ähnlich, enthält aber keine Segmentinformationen, sondern den Selektor eines Codesegmentdeskriptors. Der letztere beschreibt das Zielcodesegment.
Allein durch die Angabe eines CALL-Gate-Selektors in einem CALL FAR wird der Sprungmechanismus über das CALL-Gate ausgelöst. Da diese Befehle in ihrer Form den 8086-FAR-Sprüngen entsprechen, enthalten sie auch eine Offsetdadresse, die aber nicht benutzt wird. Die Offsetadresse des Ansprungpunktes im Zielcodesegment wird auch in das CALL-Gate eingetragen. Das CALL-Gate enthält ein Steuerbyte, in dem als wesentliche Information nur das Privilegniveau des CALL-Gates enthalten ist. Der Zugriff auf ein CALL-Gate unterliegt ebenfalls einer Privilegregel:

$$CPL \leqq DPL_{CALL\text{-}Gate}$$

wonach ein Programm nur auf CALL-Gates ein Zugriffsrecht besitzt, wenn es in bezug auf die Gates höher oder gleich priorisiert ist. Auf diese Weise kann der Privilegstufenwechsel stark eingeschränkt werden. Bild 10 illustriert die Privilegregeln bei einem Unterprogrammsprung mit Privilegstufenwechsel.
JMP FAR-Sprünge mit CALL-Gate-Selektoren sind nur für Sprünge auf die gleiche Privilegstufe möglich.

Separate Stacks

Jede Privilegstufe innerhalb einer Task besitzt ihren eigenen Stackspeicher. In der Unterprogrammtechnik – beispielsweise in der C-Compilertechnik – ist es jedoch üblich, Parameter über den Stack zu vermitteln. Bei einem Unterprogrammsprung über ein CALL-Gate werden deshalb Informationen aus dem Stack der Privilegstufe des aufrufenden Programms in den Stack der neuen Privilegstufe übertragen. Die Wortanzahl der zu übertragenden Parameter wird im CALL-Gate angegeben. In Bild 11 ist zu sehen, daß außerdem in den neuen Stack der Stackselektor, der Stackpointer, der Codesegmentselektor und der Befehlszähler des aufrufenden Programms eingetragen werden.
Bei der Abarbeitung des RET FAR-Befehls im Unterprogramm muß die Ansprungform des Unterprogramms berücksichtigt werden. Dazu vergleicht der Prozessor das Current Privileg Level (CPL) des Unterprogramms mit dem Privilegniveau des CS-Selektors der Rückkehradresse im Stack. Wenn der Rücksprung auf eine niedrigere Privilegstufe weist, wird der Stack zurückgeschrieben.

Task-Spiele

Die Hardwareunterstützung von Betriebssystemfunktionen der Multitaskverarbeitung ist ein weiteres Funktionsmerkmal des Protected Mode. Tasks sind weitgehend unabhängige Teilprogramme eines Programms, die quasiparallel abgearbeitet werden. Das geschieht durch eine sequentielle Abarbeitung mit laufender prioritätsgesteuerter Taskum-

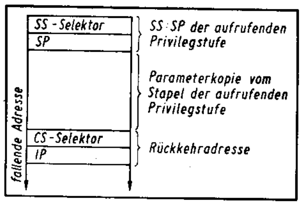


Bild 11 Stackeintragung bei Privilegebenenwechsel

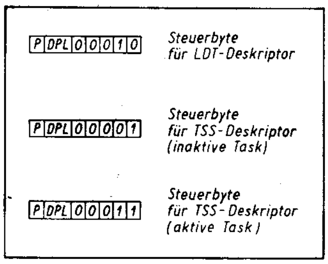


Bild 13 Steuerbytes für Systemsegmente

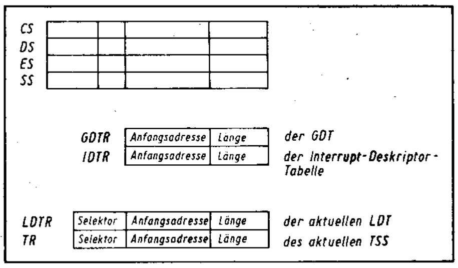


Bild 14 80601-Register für die Speicherverwaltung

schaltung. Beim 80601 beinhaltet eine Task die gesamte Programmumgebung eines Teilprogramms. Dazu gehören:
- eine eigene lokale Deskriptortabelle für die Task; die LDT ist in einem Segment enthalten, dessen Deskriptor in der GDT eingetragen sein muß
- je ein Stack für die Privilegstufen 0, 1 und 2
- ein 44-Byte-Taskstatussegment nach Bild 12 mit Platz für die Prozessorregister, die Stackpointer der drei Stacks und den Selektor der LDT.

Ein *Taskstatussegment* (TSS) besitzt in der GDT einen TSS-Deskriptor, der in Bild 13 angegeben ist. Der gesamte Vorgang einer Taskumschaltung vollzieht sich bei der Ausführung eines CALL FAR-Befehls mit der Angabe des TSS-Selektors.

Nach Bild 14 besitzt der Prozessor 80601 noch die Register TR und LDTR. Im Register TR sind der Selektor, die Anfangsadresse und die Länge des TSS der aktuellen Task abgespeichert. Bei einer Taskumschaltung werden:

- alle Taskstatusinformationen der zu unterbrechenden Task in deren TSS abgespeichert
- der Selektor des TSS-Deskriptors der neuen Task in das Register TR gesetzt
- Der Cacheteil von TR übernimmt die De-

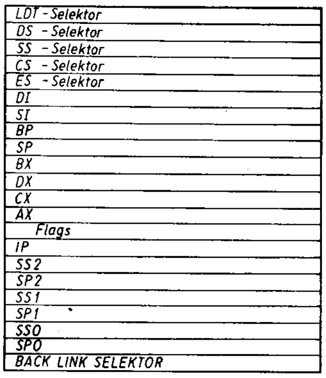


Bild 12 Aufbau des Taskstatussegmentes (TSS)

skriptorinformationen für Anfangsadresse und Länge des TSS.
- Das NT-Flag (nested task), um welches das Flagwort erweitert worden ist, wird in der TSS der neuen Task auf 1 gesetzt.
- Die Register des 80601 übernehmen die Werte aus dem TSS.
- Der Selektor der der Task zugeordneten LDT sowie deren Anfangsadresse und Länge werden in das Register LDTR geschrieben.
- In das TSS der neuen Task wird der Selektor des TSS der unterbrochenen Task eingetragen.
- Die Programmausführung beginnt in dem durch den Selektor CS im TSS vorgegebenen Codesegment an der Offsetadresse IP.

Der Taskwechsel ist die einzige Möglichkeit, auf eine Stufe niederer Priorität zu springen. Sollen in der neuen Task alle Prioritätsstufen genutzt werden, muß der Einsprung in ein Codesegment mit DPL = 3 erfolgen. Die Taskumschaltung ist nicht auf Multitaskanwendungen beschränkt. Nach dem Eintritt in den Protected Mode wird die Privilegstufe 0 eingestellt, die nur mit einem Taskwechsel verlassen werden kann.

Die Rückkehr in die unterbrochene Task ist mit dem IRET-Befehl möglich. Dieser Befehl prüft das NT-Bit im Flagregister. Im Fall der Taskverschachtelung liefert das BACK LINK SELETOR-Feld des TSS den Selektor auf das TSS der unterbrochenen Task.

Beispiel mit Taskwechsel

Das zweite Programmbeispiel enthält eine Taskumschaltung. Die anzuspringende Task 1 soll eine lokale Deskriptortabelle LDT 1 besitzen. Zunächst wird die GDT um weitere Einträge ergänzt: Der Eintrag GDT (7) in Bild 15 enthält ein CALL-Gate für den Ansprung des Ausgabeprogramms CONOUT von der Task 1 aus. GDT (8) und GDT (9) enthalten die Deskriptoren für die Taskstatussegmente der Initialtask und der Task 1. Die beiden folgenden Einträge sind der Deskriptor der lokalen Deskriptortabelle LDT 1 und eines Alias-Deskriptors für das Segment der LDT 1 mit Schreibzugriff. Der letzte Eintrag beschreibt ein separates Codesegment für Interruptprogramme.

Bild 16 zeigt die Initialisierung der lokalen Deskriptortabelle mit den Teilinformationen der Deskriptoren für Stack-, Code- und Datensegment der Task 1.

Das zweite Programmbeispiel schließt sich an die Abarbeitung des ersten Beispiels an. Das Programm beginnt im Bild 17 an der Marke main2 wieder mit dem Eintragen der Basisadressen in die GDT. Die Basisadressen innerhalb der Deskriptoren für Stack-, Code- und Datensegment der Task 1 werden in die lokale Deskriptortabelle eingetragen. Dafür wird das Alias-Datensegment für die lokale Deskriptortabelle benutzt. Nachdem das Taskregister mit dem Selektor auf den Deskriptor des Taskstatussegmentes der aktuellen Initialtask geladen worden ist, erfolgt der Wechsel zur Task 1 mit einem JMPF-Befehl. Der JMPF-Befehl enthält den Selektor auf den Deskriptor GDT (9) für das Taskstatussegment der anzuspringenden Task. Aus dem Taskstatussegment in Bild 18 ergibt sich die Startadresse der Task über den Codesegmentselektor für LDT (1) und die Offsetadresse im TSS-Feld für IP. Der Codesegmentselektor legt für das Codesegment die Privilegstufe 3 fest. Innerhalb der Task 1 in Bild 19 wird ein Unterprogramm CONOUT für eine Zeichenausgabe auf das Terminal aufgerufen. Da dieses die Privilegstufe 0 besitzt, erfolgt dieser Aufruf mit einem CALL-Gate, dessen Deskriptor in GDT (7) für das Gate die Privilegstufe 3 festlegt. Das ist notwendig, da die Task 1 nur auf Datensegmente und CALL-Gates mit der Privilegstufe 3 zugreifen darf. Das Unterprogramm benutzt privilegierte Befehle (OUT), für die ein höheres Privilegniveau eingestellt worden ist. Wichtig ist, daß sich das Privilegniveau des Unterprogramms nicht dem Privilegniveau des aufrufenden Programms anpaßt (conforming), da dann die Ausführung der privilegierten Befehle verboten ist.

Die Ausnahme ist die Regel

Die im Prozessor vorhandenen Schutzmechanismen setzen einen Mechanismus für die Behandlung von Schutzrechtsverletzungen voraus. Dafür gibt es die Behandlung von Exceptions. Exceptions und Interrupts werden mit Hilfe der Interruptdeskriptortabelle bedient. Diese enthält Interrupt- und Trap-Gates sowie Task-Gates. Interrupt- und Trap-Gates enthalten nach Bild 20 ähnlich wie ein CALL-Gate den Selektor des Codesegmentes und die Offsetadresse des Interruptprogramms. Trap-Gates lassen im Unterschied zu Interruptgates eine Interruptverschachtelung zu.

Den verschiedenen Schutzverletzungen (z. B. Segmentgrenzenüberschreitung) sind feste Positionen in der Interruptdeskriptortabelle zugeordnet. Anfangsadresse und Länge der Interruptdeskriptortabelle werden im Register IDTR des Prozessors abgespeichert.

In unserem Programmbeispiel tritt am Ende der Task 1 in Bild 21 eine Division durch 0 auf. Dafür liefert der Eintrag IDT (0) ein Interruptgate mit dem Verweis auf den Desekriptor GDT (12) für das Codesegment des Interruptprogramms mit dem Namen int0. Der IRET-Befehl führt zum Rücksprung in das unterbrochene Programm.

Literatur

/1/ Münzer, B.-G. u. a.: Mikroprozessorsystem K 1810 WM86, Teil 6. Mikroprozessortechnik, Berlin 3 (1989) 2, S. 45

/2/ Fritzsche, A.: 16-Bit-Mikroprozessor U 80601. Mikroprozessortechnik, Berlin 3 (1989) 5, S. 132

/3/ Thies, K.-D.: Die innovativen 80286/80386-Architekturen, Teil 1: 80286. Düsseldorf: te-wi-Verlag 1986

/4/ Vieillefond, C.: Programmierung des 80286. Düsseldorf: SYBEX-Verlag 1987

```
PAGE
;Global Descriptor Table - Fortsetzung

;GDT(7)          CALL GATE fuer Ausgabe
        dw       offset conout
        dw       sel_out
        db       0         ;PL 3
        db       0E4H      ;A: 1110 0100B
        dw       0
;GDT(8)          TSS: Initial-TASK
        dw       44        ;Laenge 44 Byte
b_tssi  dw       0         ;Basisadresse Bit  0-15
        db       0         ;Basisadresse Bit 16-23
        db       081H      ;A: 1000 0001B
        dw       0
;GDT(9)          TSS: TASK 1
        dw       44        ;Laenge 44 Byte
b_tss1  dw       0         ;Basisadresse Bit  0-15
        db       0         ;Basisadresse Bit 16-23
        db       081H      ;A: 1000 0001B
        dw       0
;GDT(10)         LDT fuer TASK1
        dw       80H       ;Laenge
b_ldt1  dw       0         ;Basisadresse Bit  0-15
        db       0         ;Basisadresse Bit 16-23
        db       082H      ;A: 1000 0010B
        dw       0
;GDT(11)         alias-Datensegment fuer LDT1
        dw       80H       ;Laenge
b_aldt1 dw       0         ;Basisadresse Bit  0-15
        db       0         ;Basisadresse Bit 16-23
        db       092H      ;A: 1001 0010B
        dw       0
;GDT(12)         Codesegment Interruptroutine
        dw       31H       ;Laenge
b_cexc  dw       0         ;Basisadresse Bit  0-15
        db       0         ;Basisadresse Bit 16-23
        db       09EH      ;A: 1001 1110B
        dw       0

seg_gdt          ENDS
```

Bild 15 Ergänzung der globalen Deskriptortabelle

```
PAGE
;Local Descriptor Table 1

seg_ldt1         SEGMENT PARA

;LDT1(0)         Stacksegment  PL 3
        dw       80H       ;"Boden" des Stack
b_st1   dw       0         ;Basisadresse Bit  0-15
        db       0         ;Basisadresse Bit 16-23
        db       0F6H      ;A: 11100110B
        dw       0
;LDT1(1)         Task1 Code  PL 3
        dw       100H      ;Laenge
b_ct1   dw       0         ;Basisadresse Bit  0-15
        db       0         ;Basisadresse Bit 16-23
        db       0FEH      ;A: 1111110B
        dw       0
;LDT1(2)         Task1 Daten  PL 3
        dw       100H      ;Laenge
b_dt1   dw       0         ;Basisadresse Bit  0-15
        db       0         ;Basisadresse Bit 16-23
        db       0F2H      ;A: 11110010B
        dw       0

seg_ldt1         ENDS
```

Bild 16 Lokale Deskriptortabelle für Task 1

```
PAGE
;Selektoren GDT, Fortsetzung
sel_gdt          equ      8H       ;GDT(1)
cg_conout        equ      38H      ;GDT(7)
sel_tssi         equ      40H      ;GDT(8)
sel_tss1         equ      48H      ;GDT(9)
sel_ldt1         equ      50H      ;GDT(10)
sel_aldt1        equ      58H      ;GDT(11)

;Selektoren LDT1, RPL = 3
;Selektor Datensegment hat RPL = 0
sel_s_task1      equ      7        ;LDT1(0)
sel_c_task1      equ      0FH      ;LDT1(1)
sel_d_task1      equ      14H      ;LDT1(2)

;Hauptprogramm - Fortsetzung
;Sprung zur 1. Task, Bildschirmausgabe

seg_code         SEGMENT 'init'
        ASSUME   CS:seg_code, DS:seg_gdt

;Eintrag der Segmentbasisadressen in die GDT
;-----------------------------------------
main2:  mov      ax,sel_gdt       ;Segmenadresse
        mov      ds,ax            ;laden
        mov      cl,4
        mov      ax,seg_c_exc
        shl      ax,cl
        mov      word ptr b_cexc,ax
        mov      ax,seg_tssi
        shl      ax,cl
        mov      word ptr b_tssi,ax
        mov      ax,seg_tss1
        shl      ax,cl
        mov      word ptr b_tss1,ax
        mov      ax,seg_ldt1
        shl      ax,cl
        mov      word ptr b_ldt1,ax
        mov      word ptr b_aldt1,ax

;Eintrag der Basisadressen in die LDT
;ueber ein LDT - alias - Datensegment
;----------------------------------
        ASSUME DS:seg_ldt1
        mov      ax,sel_aldt1     ;Segmentadresse
        mov      ds,ax            ;laden
        mov      ax,seg_s_task1
        shl      ax,cl
        mov      word ptr b_st1,ax
        mov      ax,seg_d_task1
        shl      ax,cl
        mov      word ptr b_dt1,ax
        mov      ax,seg_c_task1
        shl      ax,cl
        mov      word ptr b_ct1,ax
;Laden Taskregister
        mov      ax,sel_tssi
        ltr      ax
;Sprung zur 1. Task ueber jmp far
;       JMPF     TASK1
        db       0EAH
        dw       0
        dw       sel_tss1

seg_code         ENDS
```

Bild 17 Programmbeispiel mit Taskwechsel

```
PAGE
;TSS fuer Task 1

seg_tss1         SEGMENT PARA

 dw     0                        ;Back Link
 dw     100H                     ;SS PL0
 dw     sel_stack                ;SS PL 0
 dw     0                        ;SP PL 1
 dw     0                        ;SS PL 1
 dw     0                        ;SP PL 2
 dw     0                        ;SS PL 2
 dw     offset start_task1       ;IP
 dw     0020H                    ;Flags
 dw     0,0,0,0                  ;AX, CX, DX, BX
 dw     100H                     ;SP
 dw     0,0,0                    ;BP, SI DI
 dw     0                        ;ES
 dw     sel_c_task1              ;CS
 dw     sel_s_task1              ;SS
 dw     sel_d_task1              ;DS
 dw     sel_ldt1                 ;LDTR

seg_tss1         ENDS

;TSS fuer Initial-Task

seg_tssi         SEGMENT PARA

        db       44 dup('T')

seg_tssi         ENDS
```

Bild 18 Taskstatussegment für Task 1

```
PAGE
;Task1 - Stacksegment

seg_s_task1      SEGMENT STACK

        db       256 dup('S')

seg_s_task1      ENDS

;Task 1 - Codesegment

seg_c_task1      SEGMENT PARA
                 ASSUME CS:seg_c_task1, DS: seg_d_task1
```

```
;Ausgabe einer Zeichenkette auf Bildschirm
start_task1:
        mov     al,byte ptr[si]
;       CALLF   CONOUT-GATE
        db      9AH
        dw      offset conout
        dw      cg_conout
        inc     si
        cmp     al,0
        jnz     start_task1

seg_c_task1     ENDS

;Task1 - Datensegment

seg_d_task1     SEGMENT PARA

str_task1       db      0AH,0AH,0DH
                db      'Task 1 arbeitet - '
                db      'Privileg Level ist jetzt = 3'
                db      0

seg_d_task1     ENDS
```

Bild 19 Programmliste für Task 1

0000H						
P	DPL	0	0	1	Typ	
Selektor des Codesegmentdeskriptors						
Offset im Codesegment						

Bild 20 Interrupt- und Trap-Gates

✉ KONTAKT ☏

Wilhelm-Pieck-Unviersität Rostock, Sektion Technische Elektronik, Bereich Computertechnik, Albert-Einstein-Straße 2, Rostock, 2500; Tel. 40 53 87


```
PAGE
;Task 1 - Codesegment, Fortsetzung

seg_c_task1     SEGMENT PARA
                ASSUME CS:seg_c_task1, DS: seg_d_task1

        mov     bx,0
        div     bx      ;loest INT 0 aus

;Rueckkehr in Real Mode nur mit Hardware-RESET
tsk1_e: jmp     tsk1_e

seg_c_task1     ENDS

seg_c_exc       SEGMENT PARA
                ASSUME CS:seg_c_exc, SS:seg_stack
;Ausgabe einer Zeichenkette
int0:
        pusha
        mov     si,offset str_int0
int0_1: mov     al,cs:[si]
        cmp     al,0
        jz      int0_exit
;       CALLF   CONOUT-GATE
        db      9AH
        dw      offset conout
        dw      cg_conout
        inc     si
        jmp     short int0_1
int0_exit:
        popa
        iret

str_int0        db      0AH,0DH
                db      'INT 0:'
                db      'Division by Zero'
                db      0DH,0AH,0

seg_c_exc       ENDS

seg_idt         SEGMENT PARA

;Eintrag Interruptgate
        dw      offset  int0
        dw      sel_c_idt
        db      0
        db      0E6H            ;present, DPL=3

seg_idt         ENDS

                END     START
```

Bild 21 Programmbeispiel für die Behandlung der Exception 0

Mit 66 MIPS den Echtzeitproblemen zu Leibe ...

... geht der Marktführer bei Microcontrollerarchitekturen, Intel, mit dem 80960. Möglich machts die Hochintegration mit 600000 Transistoren. Was dabei herauskommt, ist eine modulare Architektur, die auf eingebettete Systeme im oberen Leistungsbereich unter dem Motto zielt: „vieles parallel – alles schnell genug – und insgesamt sehr billig". Das trifft beispielsweise auf Erkennungs- und Syntheseprobleme zu – Sprach- und Bilderkennung, Bildsynthese. Mit der Familie 960 macht Intel ein Angebot für eingebettete Rechnersysteme mit einem 32-Bit-RISC-Mikrocontroller zum halben CPU-Preis eines 486 oder eines 860, nämlich für zum Beispiel 330 $ (33-MHz-Variante, 1000er-Stückzahlen).

Bevor aber der 80960 genauer vorgestellt wird, gilt es, noch einmal Ordnung zu schaffen in den Typbezeichnungen. Also:

- Die Intel-Hauslinie heißt auch weiterhin 80386 – 80486 (künftig 80586, s. MP 9/89, S. 278). Stichwort: Mehr Leistung bei voller Stoftwarekompatibilität.
- Der 80860 ist eine völlig andere CPU, eine modulare RISC-Architekturlinie für den Bereich der Hochleistungsworkstations (Supercomputer auf einem Chip, s. MP 8/89, S. 251).
- Der 80960 ist eine 32-Bit-Mikrocontrollerlinie für eingebettete Systeme (s. MP 5/90, S. 130) in der Tradition des 8096, aber ohne Kompatibilität zu allen anderen Linien (auch nicht kompatibel zum 8096, denn das Kompatibilitätsproblem steht bei Mikrocontrollern nicht so wie bei Universal-CPUs) und wie der 80860 mit einer modularen Architektur.
- Schließlich gibt es noch den 80930, einen speziellen Hochleistungsprozessor für die im Rahmen des Joint-Ventures BiiN zwischen Intel und Siemens entwickelten ausfallsicheren Rechner auf der Basis von Multiprozessorsystemen. Der 80930 wird nicht als Einzelschaltkreis vermarktet.

Von der Familie 80960 gibt es bereits 2 Generationen:

1. Generation: 1,5-µm-CHMOS-III-Technologie. Bei 25 MHz Taktfrequenz werden maximal 20 *native MIPS* im Burstmodus erreicht. Gemeint sind natürliche Maschinenbefehle des 960 mit geringst möglicher Verzögerung durch das Speicherinterface. Umgerechnet sind das 9 VAX-MIPS und (für die Varianten mit On-chip-Gleitkommaeinheit) 1,2 MFLOPS.

2. Generation: 1-µm-CHMOS-IV-Technologie. Bei 33 MHz sind 66 *native MIPS* erreichbar, umgerechnet etwa 30 VAX-MIPS.

Diese Werte verdankt der 960 (wie auch der 860) der Parallelverarbeitung und ausgeklügelten Architekturdetaillösungen.

Die 1. Generation

Die Basisversion ist der **80960KA** (Bild 1). Er enthält den RISC-Kernprozessor, einen 512-Byte-Befehlscache (direkt abbildend) sowie einen Interruptcontroller (256 Interruptvektoren und 32 Prioritätsebenen).

Der **80960KB** ist ergänzt um eine Gleitkommaeinheit (FPU) für den Standard IEEE 754 Version 10. Diese FPU wurde gemeinsam mit dem Arithmetik-Koprozessor 387 entwik-

Bild 1 Die 1. Generation 80960

kelt (!) und ist 10mal schneller als ein 8087. 12,5 % der Chipfläche werden von der FPU belegt. Die FPU kann 32-, 64- und 80-Bit-Operanden verarbeiten. Ihr sind vier 80-Bit-Register zugeordnet (Bild 2). Der **80960MC** ist eine Spezialversion für Anwendungen im militärischen Bereich mit besonderer Eignung für die Programmiersprache ADA. Neben der FPU ist noch eine virtuelle Speicherverwaltung mit Zweiebenenzugriffsschutz integriert. Zur Adreßübersetzung gibt es einen voll assoziativen Translation Lookaside Buffer (Adreßübersetzungscache, s. MP 4/90, S. 99) für 32 Einträge bei einer Seitengröße von 4 KByte. Ein Seitenwechsel kann ohne Eingriff des Betriebssystems erfolgen. In der Hardware werden Betriebssystemfunktionen wie Taskverwaltung und Kommunikation in Multiprozessorsystemen auf der Basis von Nachrichtenaustausch (Message-Transfer) unterstützt. Für hohe Anforderungen an die Ausfallsicherheit (Fehlertoleranz) können 2 oder 4 Prozessoren 960MC zusammengeschaltet werden und sich gegenseitig überwachen. Als Spezialbaustein gibt es für solche Konfigurationen die Bus Exchange Unit (BXU) M82965. Bei 4fach-Redundanz werden je 2 Prozessoren paarweise parallel betrieben. Ein Paar ist am Bus aktiv, das andere im Hintergrund. Bei einem Fehler kann sofort zwischen den beiden Paaren von der BXU umgeschaltet werden. Die BXU ist auch in Multiprozessorsystemen einsetzbar.

Der RISC-KERN

32-Bit-Mikrocontroller werden zunehmend unter Beachtung von RISC-Prinzipien gestaltet (RISC = reduced instruction set computer, s. MP 9/88, S. 262). RISC-Kennzeichen, die auch für den 960 zutreffen, sind unter anderem eine Load-&-Store-Architektur und eine Befehlsabarbeitung in einem Takt für die Mehrzahl der Befehle. Mit der Load-&-Store-Architektur werden Speicherzugriffe von allen anderen Funktionen getrennt. Nur für ungeteilten Zugriff über den Bus gibt es für die Betriebssystemgestaltung (Ressourcenverriegelung mit Semaphoren) atomare, das heißt unteilbare Operationen. Beim 960 sind das ein Bitmaskierungs- und ein Additionsbefehl.

Ein weiteres RISC-Merkmal wurde beim 960 nur bedingt erfüllt: Der Prozessor ist teilweise mikroprogrammiert. (RISC im strengen Sinne bedeutet Verzicht auf Mikroprogrammierung. RISC-Prinzipien wurden im 960 in dem Maße umgesetzt, wie es dem Architekturentwurfsziel zuträglich war.) Die einfachen Befehle (Speicherzugriff, Verarbeitungsfunktionen, Steuerfunktionen) sind im RISC-Kern „direkt verdrahtet" und werden meist in einem Takt abgearbeitet. Komplizierte Befehle (z. B. CALL/RETURN, Systeminitialisierung, Interrupt- und Fehlerbehandlung, CPU-Selbsttest, Cacheverwaltung) sind in Form von Mikroprogrammen realisiert, die in einem Mikroprogramm-ROM der Organisation 3 K × 42 Bit stehen. Die Mikroprogramme arbeiten mit dem Befehlssatz des RISC-Kerns. Das ist eine Besonderheit des 960: Der Mikrobefehlssatz ist zugleich Teil des Befehlssatzes für die Anwenderprogrammierung. Der 960 folgt damit dem von Hewlett-Pakkard-RISC-Prozessoren sowie aus Fadencodesystemen bekannten Millicodekonzept, wobei der Nutzer den Mikrobefehlssatz des Kerns (im Unterschied z. B. zum mikroprogrammierten 8086) direkt nutzen kann. Im Befehlssatz wird durch einen CALL-/RETURN-Mechanismus eine parallele Ausführung von Verarbeitungsbefehlen und Millicode-Ruf- und -Rückkehrbefehlen unterstützt. Im Idealfall entsteht somit praktisch kein Overhead bei der Abarbeitung einer Millicodesequenz im Unterschied zu einer Sequenz von Kernbefehlen.

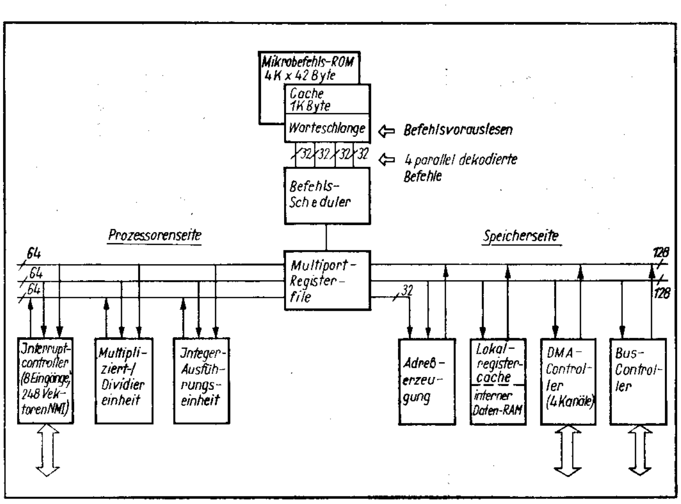
Dieses Konzept unterstützt die modulare Erweiterungsfähigkeit der Architektur auf Hardware-, Mikroprogramm- und Anwendersoftwareebene. Bei der Erweiterung des 960 um Zusatzeinheiten bleibt die Kompatibilität auf einfachere Weise als in der 8086er Linie gesichert. (Die Wahrung der Kompatibilität vom 8086 bis zum 486 erforderte beträchtliche Hardwareklimmzüge, was sich in einer sehr komplizierten internen Chipgestaltung, hohen Entwicklungszeiten und -kosten sowie letztlich in Architekturballast widerspiegelt.)

Das Millicodekonzept wirkt dem Engpaß „Busschnittstelle" entgegen. Durch die höhere Codedichte von mikroprogrammierten Befehlen in Verbindung mit einem On-chip-Cache läßt sich der Durchsatz steigern.

Dem RISC-Kern sind mehrere 32-Bit-Registersätze zugeordnet. Ein Anwenderprogramm „sieht" 16 globale Register für die Übergabe von maximal 13 Parametern zwischen Prozeduren, 16 lokale Register für 13 Daten, die in der Prozedur benötigt werden, sowie 32 Spezialfunktionsregister (SFR, s. MP 5/90, S. 130) zur Hardwaresteuerung (z. B. Steuerung des Verhaltens der Busschnittstelle oder des Interruptcontrollers). Die restlichen globalen und lokalen Register werden zur Programmsteuerung genutzt. Außerdem gibt es noch 32 nur intern genutzte Pufferregister.

Die 16 lokalen Register sind als Fenster in einem Registerstack-Cache gebunden. In der Grundversion hat der Registerstack Platz für 4 Registersätze. Bei Prozeduraufruf wird einfach ein neues Fenster eröffnet, so daß keine Datenrettung betrieben werden muß. Programmkontextumschaltungen werden somit extrem schnell. Erst wenn alle Fenster „ausgebucht" sind (also bei Nestung von 4 Unterprogrammen), müssen Daten in den Externspeicher ausgelagert werden. Dazu ist der Blocktransfermodus der Busschnittstelle nutzbar (Blockgröße 16 Byte, Bustransferrate bei 20 MHz Taktfrequenz 53 MByte/s). Wenn Prozeduren keine lokalen Registerzugriffe benötigen, kann auch ein Prozeduraufruf ohne Eröffnung eines neuen Fensters programmiert werden (Befehl BRANCH-_&_LINK).

Praktische Erfahrungen besagen, daß nur in der Startphase von C-Programmen größere Nestungstiefen (z. B. um 10 Ebenen) benötigt werden. „Mitten drin" kommt man mit 4 Ebenen schon so weit, daß Datenrettung in den Externspeicher selten ist. Der Befehlssatz unterstützt die 3-Operanden-Adressierung (2 Quelloperanden und 1 Zieloperand). Eine spezielle Hardwaresteuerung auf dem Chip ermöglicht die parallele Abarbeitung mehrerer Operationen. Um Zugriffskonflikte auf gemeinsam benutzte Ressourcen auszu-



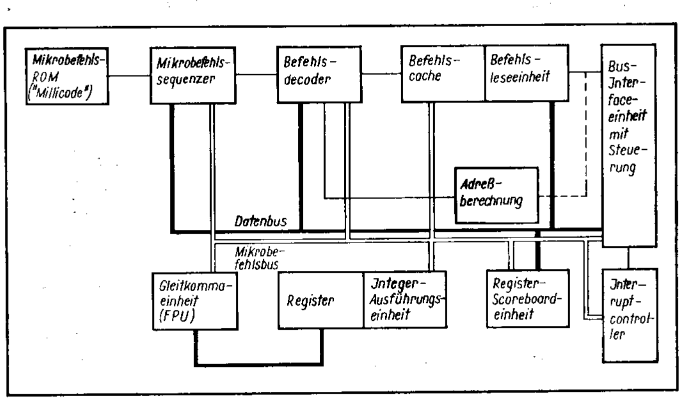


Bild 3 Der 80960CA

◀ *Bild 2 Der 80960KB*

schließen, wurde ein „Scoreboarding“ für Register und für Bedingungscodeflags vorgesehen (englisch: scoreboard, im Sport: Trefferanzeigetafel). Ist eine Ressource durch eine laufende Operation belegt, wird der Zugriff durch ein gesetztes Bit verriegelt. Dieser Verwaltungsmechnismus kostete 1 % Chipfläche, bewirkte aber durch die Parallelisierung von Befehlen eine Leistungssteigerung um 20 bis 30 %. Die Hardware-Zugriffsverriegelung hat Intel der Architektur von Großrechnern entlehnt.
Das in manchen RISC-Architekturen übliche Verriegeln auf Softwareebene durch einen intelligenten Compiler war beim 960 nicht anwendbar, da für Echtzeitanwendungen die Assemblerprogrammierung (wenigstens ausgewählter Passagen) unverzichtbar bleibt. Schließlich dürfen beispielsweise in einem Navigationssystem (englisch: mission critical applications, Anwendungen in Luft- und Raumfahrt) nicht eines Tages durch einen Programmierfehler bei Strafe des Absturzes Havariesituationen heraufbeschworen werden ...

2. Generation

Mit der Erhöhung des Integrationsgrades bei CHMOS-IV (daraus resultiert bereits eine technologisch bedingte Erhöhung der Leistungsfähigkeit bei gleicher Architektur) baute Intel die Parallelisierungskonzepte weiter aus (Bild 37). 80960CA heißt der erste Repräsentant, für den die Superskalartechnologie angewendet wurde, mit der die gleichzeitige Abarbeitung von bis zu 3 Befehlen möglich wird. Die Steuerung übernimmt der Hardware-Scheduler. Neben der Parallelisierung auf der Ebene der Prozessoren (also FPU, Integer-RISC-Kern) wurde auch im Kern stärker parallelisiert. Der Befehlsstrom wird in 3 Befehlsklassen parallel abgearbeitet:
– Register-Register-Operanden (arithmetische, logische und Bitoperationen)
– Speicherzugriffsoperationen
– Programmablaufsteuerung (Verzweigungsbefehle).
Hinter jeder Befehlsklasse steht in der Hardwarerealisierung eine Pipeline, in der der Teilbefehlsstrom nach dem Fließbandprinzip an mehreren Stationen ausgeführt wird.
Beim Einlesen des Befehlsstromes vom Speicher werden stets 4 Befehle gemeinsam in einen 128 Bit breiten Befehlscache übernommen und nach Möglichkeit parallel ausgeführt. Während zum Beispiel Arithmetikbefehle abgearbeitet werden, können über das Businterface bereits die nächsten Daten eingelesen werden.
Da RISC-Prozessoren je Befehl nur 1 bis 2 Taktzyklen brauchen, kann man mit dem 960 bei 33 MHz theoretisch 99 native MIPS erzielen. Praktisch ist eine solche Parallelisierung nicht real, so daß man „nur“ auf die in der Überschrift genannten 66 native MIPS kommt – das entspricht 2 Befehlen je Taktzyklus. Für den schnellen Datentransfer zwischen den Pipelines und den Registern verfügt das Registerfile über 6 Datenports. Ergebnisse eines Verarbeitungsschrittes, die sofort wieder gebraucht werden, werden in das gewählte Zielregister eingetragen und gleichzeitig intern der Ausführungseinheit unverzögert wieder zur Verfügung gestellt. Damit braucht nicht erst gewartet zu werden, bis das Ergebnis im Register steht und erneut ausgelesen ist. Das ist eine typische RISC-Technik (Register-Bypass).
Neben dieser Parallelisierung im Prozessorkern gibt es noch die Parallelität auf Prozessorebene. Der 960CA ist ein Multiprozessorsystem mit folgenden Controllern neben der CPU:
– **DMA-Controller** (mit 4 unabhängigen Kanälen, die mit festen oder rotierenden Prioritäten arbeiten können). Dieser ist in 3 Betriebsarten programmierbar (Blockmodus für Transfer zwischen Speicherbereichen, Demand-Modus für Transfer zwischen Peripherieeinheiten und Speicher sowie Fly-by-Modus für extrem schnelle Übertragung ähnlich dem Blockmodus, aber ohne interne Zwischenpufferung im Prozessor und mit nur einer Busanforderung). Im Fly-by-Modus sind Transferraten von bis zu 59 MByte/s bei einer Taktfrequenz von 33 MHz erreichbar. DMA-Übertragungen können parallel zur Programmausführung laufen. Die Reaktionszeiten bis zur Bedienung eines DMA-Kanals können etwa 400 ns betragen.
– **Buscontroller** (DMA-fähig, Blockübertragung im Burstmodus). Der 4-GByte-Adreßraum ist in 16 Segmente zu je 256 MByte strukturiert. Für jedes Segment kann ein spezifisches Übertragungsregime vereinbart werden. So kann man die Datenbusbreite programmieren (8, 16 oder 32 Bit), das Übertragungsprotokoll (Wartezyklensteuerung, Ready-Steuerung, Adreßpipelining) oder die Datenrepräsentation (Little Endian oder Big Endian). Diese Architekturdetails sind bereits vom 860 und vom 486 bekannt (s. MP 8/89, S. 251 und MP 9/89, S. 278). Der Buscontroller wird von DMA-Controller und CPU gemeinsam genutzt.
– **Interruptcontroller** (248 externe Quellen, 32 Prioritätsstufen). Der Interruptcontroller ist flexibel programmierbar und läßt sich so an verschiedene Interruptkonzepte (z. B. MULTIBUS-Interruptsystem) anpassen. Als Interruptreaktionszeiten werden bei einer Taktfrequenz von 33 MHz 750 ns für den NMI und 900 ns für die restlichen Interrupteingänge angegeben.
Um den Flaschenhals „Busschnittstelle“ zusätzlich aufzuweiten, enthält der 960CA folgende Cacheformen:
– einen direkt abbildenden Befehlscache von 1 KByte
– einen statischen Daten-RAM, 1,5 KByte, 128 Bit breit, konfigurationsabhängig nutzbar als Datencache, als Speicher für globale Daten und als Multi-Registerfile (5 bis maximal 15 Registerfenster sind definierbar).
Der RISC-Kern hat für bestimmte Verarbeitungsfunktionen getrennte Ausführungseinheiten, so eine Integer-ALU, eine Multiplizier-/Dividiereinheit und eine Adreßarithmetik.

Als Systemhaus kann es sich Intel natürlich nicht leisten, dem Anwender nur einen nackten Prozessorchip anzubieten. Wer an Qualität und Leistungsfähigkeit gewöhnt ist, wird bei Markteinführung eines neuen Produkts entsprechende Unterstützungsleistungen erwarten. Beim 80960 sind das der LAN-Koprozessor 82596CA, schnelle 7,5-ns-EPLD-Adreßdecoder (85C508), Chips für das Burstmodus-Interface (85C960 mit Wartezyklengenerator, Ready-Steuerung, Adreßdecoder), sowie 1-MBit-EPROMs 27C960 mit Unterstützung für Burstzugriff bis 33 MHz ohne Wartezyklen. Shadow-RAMs, in die der EPROM-Inhalt beim Systemstart kopiert wird, kann man so einsparen.
Für die Softwareentwicklung gibt es C-Compiler (ANSI-Standarderweiterungen und Gleitkommaunterstützung) sowie einen leistungsfähigen Makroassembler. Die Codeoptimierung läßt sich für maximalen Durchsatz oder für minimalen Speicherbedarf vornehmen. Für den Systemtest sowie zur Leistungsbewertung der Anwendersoftware stehen ein Quellcodedebugger sowie Simulations-/Analyseprogramme zur Verfügung.

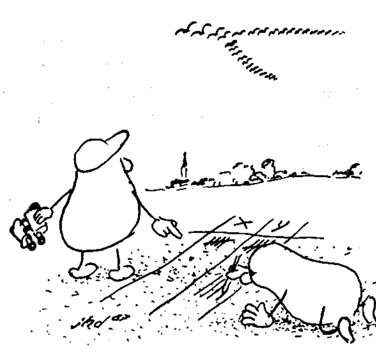
Frisch von der CeBIT '90 ein Anwendungsbeispiel: Im Newgen TurboPS/480, einem 800 × 400-dpi-Postscript-Laserdrucker, arbeiten ein 20-MHz-80960KB als Grafikprozessor und ein 80186 als I/O-Prozessor zusammen. Sehr zur Freude des Anwenders, der hohe Auflösung und hohe Ausgabegeschwindigkeit in einem Gerät vereint findet.

PS: Immer nur Intel!?
Intel wird in Irland in der Nähe von Dublin mit staatlicher Förderung drei neue Fabriken errichten. Damit gibt die kalifornische Firma der Perspektive materielle Gewalt, bei Systemlösungen in Europa von einem Markanteil von zur Zeit 24 % auf 43 % im Jahre 1993 zu kommen. Da Irland EG-Mitglied ist, hat der Erfinder des Mikroprozessors somit bereits ein Standbein auf dem europäischen Binnenmarkt ab 1992 ...

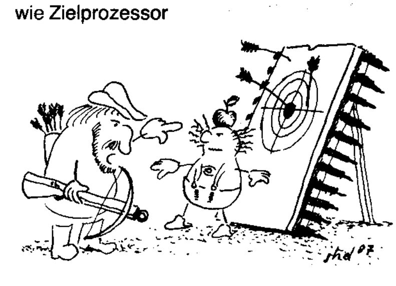
guv

Kleines Lexikon der Mikrorechentechnik

X und **Y**
wie X, Y-Register



Z
wie Zielprozessor



Zeichnungen: Dahmen

Der Gate-Array-Schaltkreis U 5303 FC303

Günter Sedlacek, Hartmut Lubrich, Jena

Der Gate-Array-Schaltkreis U 5303 FC303 ist ein Zusatzschaltkreis für den Floppy-Controllerschaltkreis U 8272, mit dem der Aufbau von kompletten Floppy-Controllern mit nur 4 Schaltkreisen möglich wird.

Der Schaltkreis U 5303 FC303 (FDC_ZUS) wurde für den Aufbau von Floppy-Disk-Controllern auf der Basis des LSI-Schaltkreises U 8272 entworfen. Er integriert folgende Baugruppen (siehe Bild 1):
- Takt-/Daten-Separator mit digitaler PLL (PLL)
- Schreibtaktgenerator und Schreibsteuerung mit Präkompensationslogik (CLK_WRC)
- Decoder zur Laufwerkauswahl, Controller-Steuerregister für Bitratenumschaltung und den Anschluß an einen DMA-Schaltkreis (DMAINT_CONT).

Mit diesem ASIC wird der Aufbau eines kompletten Floppy-Disk-Controllers mit 4 Schaltkreisen und 1 bzw. 2 Taktgeneratoren (16/9,6 MHz) möglich (U 5303 FC303, U 8272, 2 × 74S240), der direkt auf einer Zentraleinheitsplatine angeordnet werden kann und den System-DMA-Schaltkreis für seine Übertragung nutzt. Es sind alle üblichen Laufwerke mit FM- und MFM-Codierung sowie Bitraten von 250, 300 und 500 kHz anschließbar. Die maximal mögliche Taktfrequenz beträgt 20 MHz.

Schaltungsbeschreibung

Die Aufteilung in Baugruppen, deren Funktion im folgenden erläutert werden soll, ist in Bild 1 dargestellt.

Controller-Steuerregister- und Anschluß-DMA-Schaltung

Die Baugruppe DMAINT_CONT setzt sich aus 2 Registern und der entsprechenden Ansteuer- und Ausgabelogik zusammen. Die Register werden vom Mikrorechner mit normalen E/A-Befehlen beschrieben. Das Register 1 arbeitet als Steuerregister. Register 2 dient der Bitrateneinstellung. Es sind folgende Funktionen möglich:
- Laufwerksauswahl und Schalten der Motoren
- Abfrage der Laufwerksbereitschaft (RDY)
- Erzeugung des Chip-Select und eines programmierten Resets für den U 8272
- Enable DMA und Interrupt
- Verkettung der DMA-Übertragungen nach Adressenüberlauf.

Die Steuerung kann über die Leitung /RMX auf einen von 2 Modi eingestellt werden.

RMX-Modus (/RMX = Low)
In diesem Modus erfolgt die Laufwerksauswahl und die Auswertung der RDY-Meldung durch den U 8272. Es sind 4 Laufwerke wählbar. 2 DMA-Kanäle können verkettet werden.

IBM-Modus (/RMX = High)
Dieser Modus ist kompatibel zu dem des IBM PC/AT. Die Auswahl von 2 Laufwerken erfolgt über ein Bit des Steuerregisters. Das Bereitschaftssignal RDY wird mit einem Eingabebefehl abgefragt und von der ZVE ausgewertet. Es erfolgt keine Verkettung von DMA-Kanälen.

Takterzeugung/Schreibsteuerung

Die Baugruppe CLK_WRC beinhaltet die Systemtakterzeugung für den Schaltkreis, den Taktgenerator für den Floppy-Controller, den Schreibtaktgenerator sowie die Schreibsteuerung. Die Systemtakte für das Gate-Array werden aus den Eingangstakten CLK0 (16 MHz) und CLK1 (9,6 MHz) entsprechend der eingestellten Bitrate ausgewählt. Aus dem Systemtakt CA wird mit programmierbaren Teilern der Takt CFDC für den Floppy-Controller-Schaltkreis und der Schreibtakt WRC gebildet. Tafel 1 zeigt den Zusammenhang zwischen der im Steuerregister eingestellten Bitrate und den einzelnen Takten.
Aus der Tafel 1 ist ersichtlich, daß der zweite Taktgenerator CLK1 (9,6 MHz) nur für die Bitraten 150 und 300 kHz notwendig ist. Die Impulsbreite des Schreibtaktes beträgt in Abhängigkeit von der eingestellten Bitrate 125 ns, 208,33 ns oder 250 ns. Die Schreibsteuerung erzeugt aus den vom U 8272 gelieferten Schreibimpulsen WRD und den Steuersignalen PS0 und PS1 die präkompensierten Schreibimpulse für das Laufwerk (siehe Tafel 2). Die Schreibkompensation beträgt 1/16 des Schreibtaktes.

Tafel 1 Zusammenhang zwischen Bitrate und Taktfrequenzen

Bitrate (Register)	CA in MHz	Bitrate in kHz FM	MFM	CFDC in MHz	WRC in kHz FM	MFM
00	16	250	500	8	500	1000
01	9,6	150	300	4,8	300	600
10	16	125	250	4	250	500

Tafel 2 Zusammenhang Präkompensationssignale und Bildung der Schreibimpulse WD

WE	ENPS	PS0	PS1	Schreibimpulse
0	·	·	·	keine Schreibimpulse
1	0	*	*	Normallage
1	1	0	0	Normallage
1	1	0	1	vorgezogen 1/16 Takt
1	1	1	0	verzögert 1/16 Takt
1	1	1	1	Normallage

Takt-/Daten-Separator (PLL)

Der Takt-/Daten-Separator ist eine digitale PLL, wobei ein einstellbarer 4-Bit-Zähler als steuerbarer Frequenzgenerator verwendet wird. Der Phasenvergleich erfolgt durch Abfragen der Zählerstellung gleichzeitig mit den einlaufenden Datenimpulsen und durch den Vergleich mit der Soll-Zählerstellung 8.
Nach jedem Leseimpuls erfolgt eine Korrektur des Zählers um maximal ± 2 Impulse. In Abhängigkeit vom Abstand der 3 zuletzt eingelaufenen Datenimpulse voneinander (Kennzeichen der Folge) wird der Korrekturwert entweder aus der Ablage des vorletzten Impulses, aus der Summe der Ablagen des vorletzten und des davorliegenden Impulses oder aus der Frequenzabweichung über die letzten 7 Impulse gewonnen. Dementsprechend enthält die Schaltung eine Pipeline zum Speichern der letzten Ist-Zählerstände, einen Adder zur Addition zweier Werte, einen Zähler zur Bestimmung der Frequenzabweichung über die letzten 7 Impulse, ein Schieberegister zur Ermittlung des Kennzeichens der aktuellen Folge und die kombinatorische Logik und Steuerung zur Bereitstellung des Korrekturwertes aus diesen Informationen.
Die Datenimpulse für den U 8272 werden von der Vorderflanke der Leseimpulse abgeleitet; das Window-Signal liefert ein dem steuerbaren Frequenzgenerator nachgeschaltetes Teiler-Flipflop.
Die digitale PLL ist mit dem Signal VCO des U 8272 schaltbar. Der erste auf dieses Signal folgende Datenimpuls startet die PLL mit der Soll-Phasenlage. Mit dem realisierten Regelalgorithmus sind FM- und MFM-Datenfolgen bis zu einer Frequenzabweichung von ± 2,3 % und einer Ablage der Datenimpulse (Peakshift) von 30 % gegenüber der Ideallage fehlerfrei lesbar.

Ein- und Ausgänge

Im Bild 2 ist das Schaltzeichen des U 5303 FC303 dargestellt. Die Signalbezeichnungen haben die in den Tafeln 3 und 4 aufgelistete Bedeutung. Die Abkürzungen in der Spalte Pintyp haben folgende Bedeutung:
E – Eingang

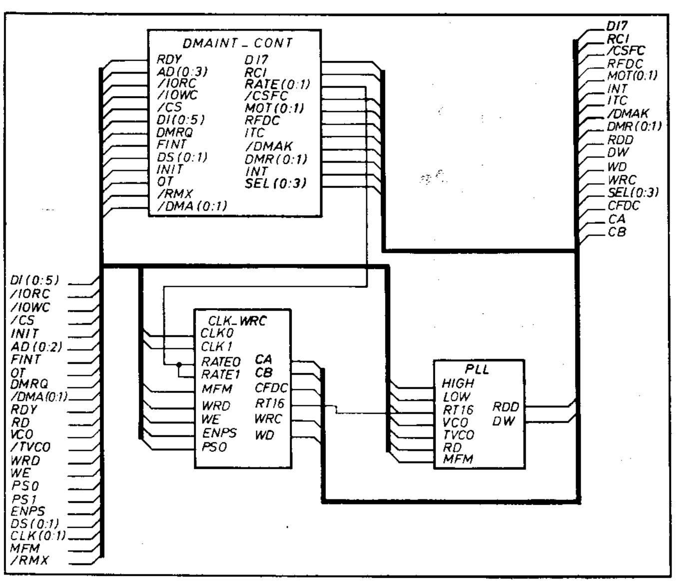


Bild 1 Blockschaltbild U 5303 FC303

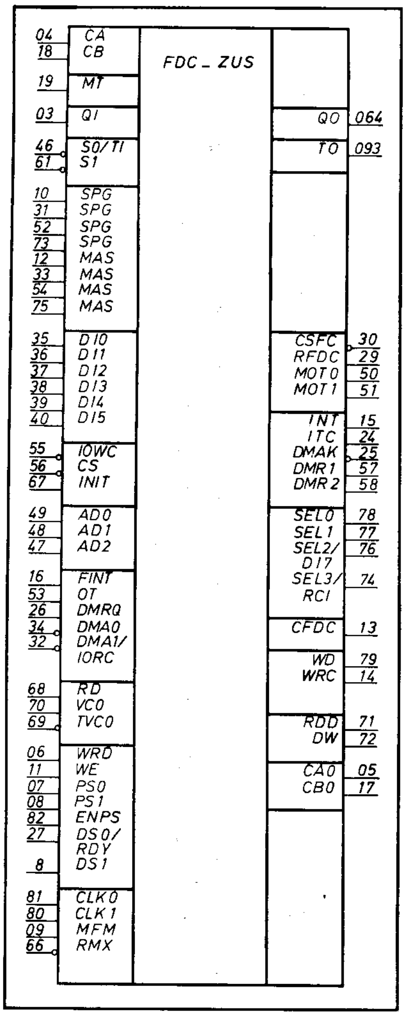


Bild 2 Schaltzeichen des U 5303 FC303

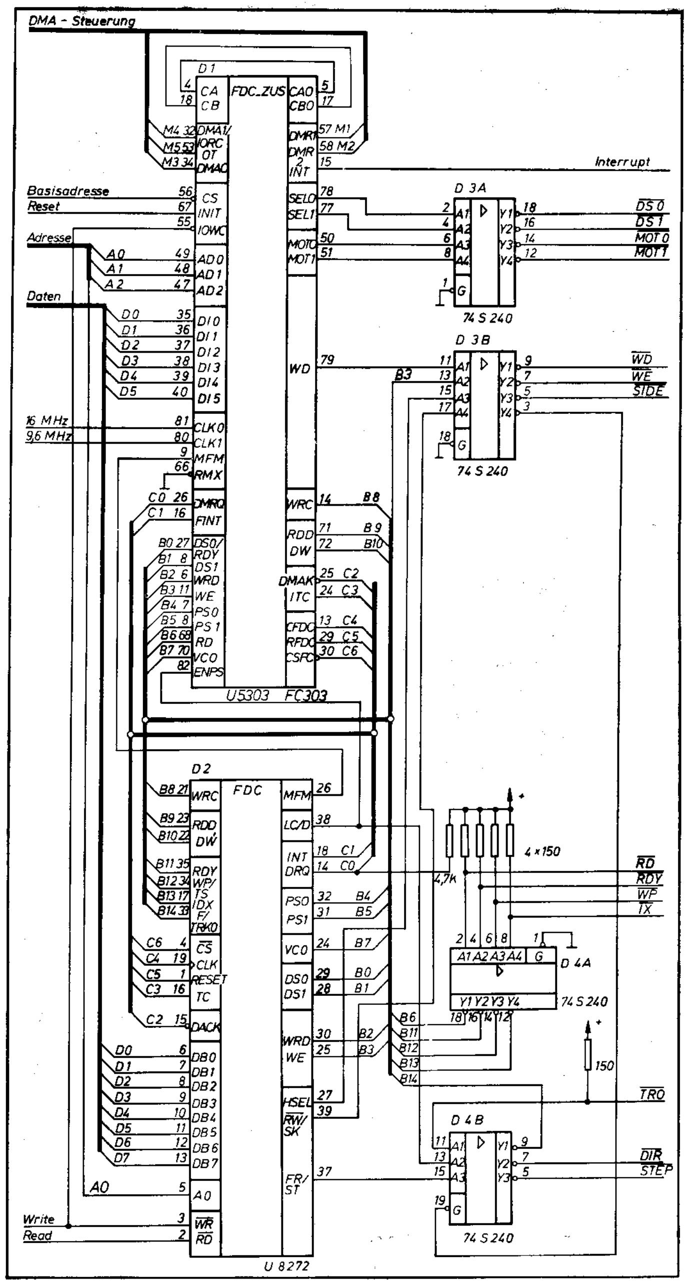


Bild 3 Controller im RMX-Modus

Tafel 3 Systempins des Schaltkreises U 5303 FC303

Pin-name	Pintyp	Erläuterungen
MAS	S	Masseanschluß
SPG	S	Betriebsspannung
CA	E (LHT)	Takteingang A
CB	E (LHT)	Takteingang B
MT	E (LHT)	Eingang Modussteuersignal
QI	E (LHT)	Eingang LSSD-Schiebekette
QO	A (GT)	Ausgang LSSD-Schiebekette
S1	E (LHT)	Interfacesteuersignal 1
SQ/TI	E (LHT)	Interfacesteuersignal 0/ Eingang der Teststruktur
TO	A (GT)	Ausgang der Teststruktur

Tafel 4 Nutzerpins des Gate-Array-Schaltkreises U 5303 FC303

Pin-name	Pintyp	Erläuterungen
CAO	A (GT)	Systemtakt A
WRD	EA (LHT)	Schreibdaten vom U 8272
PS0	E (LHT)	Schreibdaten verzögern vom U 8272
PS1	E (LHT)	Schreibdaten vorziehen vom U 8272
MFM	E (LHT)	Umschalten auf MFM vom U 8272
WE	E (LHT)	Write-Enable vom U 8272
CFDC	A (GT)	Takt für U 8272
WRC	A (GT)	Schreibtakt für U 8272
INT	A (GT)	Interrupt für CPU
FINT	E (LHT)	Interrupt von U 8272
CBO	A (GT)	Systemtakt B
ITC	A (GT)	Time-out-Signal für U 8272
DMAK	A (GT)	DMA-Acknowledge für U 8272
DMRQ	E (LHT)	DMA-Request vom U 8272
DS0/ RDY	E (LHT)	Drive-Select 0 vom U 8272 (im RMX-Modus)/Laufwerk bereit (im IBM-Modus)
DS1	E (LHT)	Drive-Select 1 vom U 8272
RFDC	A (GT)	Reset U 8272
CSFC	A (GT)	Chip-Select U 8272
DMA1/ IORC	EA (HHT)	Acknowledge DMA-Kanal 2 (im RMX-Modus)/Leseimpuls (im IBM-Modus)
DMA0	E (HHT)	Acknowledge DMA-Kanal 1
DIO	E (LHT)	Eingabedaten Steuerregister
DI1	E (LHT)	wie DI0
DI2	E (LHT)	wie DI1
DI3	E (LHT)	wie DI2
DI4	E (LHT)	wie DI3
DI5	E (LHT)	wie DI4
AD2	E (LHT)	Auswahladresse
AD1	E (LHT)	wie AD2
AD0	E (LHT)	wie AD1
MOT0	A (GT)	Motor on für Laufwerk 0
MOT1	A (GT)	Motor on für Laufwerk 1
OT	E (LHT)	Time-out-Signal des DMA
IOWC	E (HHT)	Schreibimpuls
CS	E (HHT)	Controller-Select (Basisadresse)
DMR1	A (GT)	Request an DMA-Kanal 1
DMR2	A (GT)	Request an DMA-Kanal 2
RMX	E (HHT)	Umschaltung RMX-/IBM-Modus
INIT	E (LHT)	Systemreset
RD	EA (LHT)	Lesedaten vom Laufwerk
TVCO	EA (HHT)	Test-VCO-Enable
VCO	EA (LHT)	VCO-Enable vom U 8272
RDD	A (GT)	Lesedaten für U 8272
DW	A (GT)	Datenfenster für U 8272
SEL3/ RC1	A (GT)	Drive-4-select-Bit 4 (im RMX-Modus) Lesen Identifikation (im RMX-Modus)
SEL2/ DI7	A (GT)	Drive-3-select-Bit 3 (im RMX-Modus) RDY-Abfrage (im IBM-Modus)
SEL1	A (GT)	Drive-2-select-Bit 2
SEL0	A (GT)	Drive-1-select-Bit 1
WD	A (GT)	Schreibdaten für Treiber
CLK1	E (LHT)	Takteingang 2 (9,6 MHz)
CLK0	E (LHT)	Takteingang 1 (16 MHz)
ENPS	E (LHT)	Enable Präkompensation (LC/DR des U 8272

EA – Eingang mit Latch auf Systemtakt CA
EB – Eingang mit Latch auf Systemtakt CB
LHT – Eingangsstufe mit Low-Haltetransistor
HHT – Eingangsstufe mit High-Haltetransistor
A – Ausgang
GT – Gegentaktausgangsstufe
S – Spannungsversorgung

Applikationsbeispiel

Bild 3 zeigt die Applikation des Schaltkreises in einem Floppy-Controller zur Ansteuerung von 2 Laufwerken. Es wurde der RMX-Modus verwendet, da dieser auch mit den üblichen Laufwerken eine Signalisierung von Diskettenwechseln über den U 8272 erlaubt. Als Treiber werden zwei 74S240 eingesetzt, die außer der notwendigen Treiberleistung das Demultiplexen des Steuer- und Statussignals des U 8272 gestatten. Außer den normalen Signalen der CPU und des DMA-Schaltkreises sind die Takte und eine dekodierte Basisadresse bereitzustellen (Adreßbits 3 – 15 dekodiert). Der Takt 9,6 MHz ist nur für die Bitraten 150 und 300 kHz.

Der Commodore Amiga

Thomas Bölkow, Berlin

Der Amiga 500 hat schon in der Grundausstattung eine Speicherkapazität von insgesamt 0,75 MByte (0,5 MByte Arbeitsspeicher/256 KByte Betriebssystem) und kann mit einer Steckkarte einfach um 512 KByte erweitert werden. Ein 3,5-Zoll-Mikro-Diskettenlaufwerk mit formatierter Kapazität von 880 KByte ist zusammen mit einer großzügigen Volltastatur in einem Gehäuse integriert. Anstelle des Amiga-Farbmonitors kann über einen zusätzlichen HF-Modulator auch ein Farbfernseher angeschlossen werden. Der Amiga 500 arbeitet mit dem Motorola-Prozessor 68000, der von drei weiteren eigenständigen Prozessoren für Grafik, internes Datenmanagement und für Kommunikation unterstützt wird.

Mit diesem Beitrag soll nach dem Artikel über den Apple Macintosh in MP 10/89, S. 308 ein weiteres alternatives PC-Konzept vorgestellt werden. Es wird besonders jetzt mehr Augenmerk auf die Rechner mit Prozessoren der 68000er-Familie, sowohl von Atari als auch von Commodore, zu legen sein: Nach einer Prognose des Markforschungsinstituts IDC wird der Anteil der 68000er bis 1992 auf 26,2 % anwachsen, heute beträgt er schon 13 %, die MS-DOS-Kompatiblen sollen bei 46,9 %, UNIX-Rechner bei 5 % und das Apple-Mac-Konzept bei 8 % liegen.
Für diese Entwicklung sind schon heute alle Weichen gestellt. Die Vorstellung des Atari Mega ST und des Commodore A3000 sind in dieser Hinsicht richtungweisend. Auch hier verschwindet wie bei den größeren MS-DOS-Kompatiblen (ab i80386) die Grenze zwischen PC und Workstation.

Die Produktpalette

Der erste Amiga wurde im Jahre 1985 vorgestellt. Es war der Amiga 1000. Damals mußte das Betriebssystem noch von Diskette eingeladen werden, und der RAM betrug standardmäßig nur 256 KByte. Wegen seiner Grafikeigenschaften, die die eines normalen PCs bei weitem überflügelten, erhielt er einen Achtungserfolg, konnte jedoch wegen seines hohen Preises keine weite Verbreitung erlangen. Die kam erst im Jahre 1987 mit dem Amiga 500, der eindeutig auf den großen Bereich der Heimanwender und der Schulen und Universitäten abzielte (siehe Bild oben).
Der Erfolg war vorprogrammiert, denn für den A500, der als Nachfolger des legenderen Commodore 64 galt, sprachen unter anderem sein geringer Preis und einige Verbesserungen gegenüber dem A1000, wie sein kleineres Gehäuse, standardmäßig 512 KByte RAM und ein fest im ROM installiertes Betriebssystem.
Auch für den Büroanwender wurde 1987 eine Lösung angeboten – der Amiga 2000 – mit 1 MByte Hauptspeicher und im klassischen PC-Gehäuse, also mit viel Platz für Erweiterungskarten.
Mit dem Amiga 2500 stieß man schließlich in den Bereich der DTP-Computer vor, und in Zukunft soll mit dem Amiga 3000 professionelle 3D-Animation bei kurzen Rechenzeiten möglich werden.

Das Konzept

Der Amiga 1000 hat ein den PCs ähnliches Gehäuse. Die Bauform des Amiga 2000 und des A 2500 ist das Standard-Desktop-Gehäuse (siehe Bild S. 174). Der 500er dagegen hat in etwa die Größe eines Keyboards und ist somit wie gemacht für den Hobbybereich, und kann um ein zigarrenkistengroßes Zweitlaufwerk erweitert werden.
Die Benutzeroberfläche – die Workbench – könnte man wie beim Mac auch als Human Interface bezeichnen, denn hier hat es der Anwender ebensowenig nötig, Kopfstände bei der ersten Begegnung mit dem System zu vollführen. Durch die mitgelieferte Lernsoftware ist das eher unterhaltsam als anstrengend. Wer allerdings später bestimmte Systemabläufe programmieren möchte, muß auf das spartanische CLI (Command Line Interface) zurückgreifen.
Einige repräsentative Demos finden Sie auf der zweiten Umschlagseite.

Die Hardware

Der Amiga ist prädestiniert für den professionellen Büroeinsatz, besonders im Grafik- und Animationsbereich sowie als preiswertes DTP-System. Damit ist eine gewisse Abgrenzung zum PC-Sektor gegeben, dessen Domäne ja hauptsächlich der Text- und Datenbankbereich ist.
Die CPU des Amiga ist in den meisten Fällen ein MC 68000, seit 1989 beim A2500 und beim A3000 ein MC 68020 bzw. MC 68030. Die 3 ASICs des Amiga garantieren optimale Eigenschaften. Diese 3 Spezialchips, auch Customchips genannt, wurden eigens von Commodore für den Amiga entwickelt.
Der Grafik- und Animationsship (Blitter) ermöglicht einen sehr schnellen Datentransfer zwischen bis zu 3 Quellen. Er kann bis zu 1 Millionen Punkte pro Sekunde auf dem Bildschirm darstellen und kontrolliert 25 DMA-Kanäle (Direct Memory Access).
Der Videochip sorgt für die Darstellung verschiedener Auflösungen, von 320 × 200 bis 640 × 512 Punkten einschließlich des Overscanformats mit 768 × 580 Punkten (der gesamte Bildschirm wird genutzt). Die Normaleinstellung, in der beispielsweise die Workbench arbeitet – nutzt 640 × 200 Punkte. Eine negative Erscheinung dabei ist, daß bei einer Zeilenzahl ab 400 die Bildwiederholrate scheinbar auf 25 Hz gemindert wird (Interlacemodus), das heißt, es wird immer nur das obere oder das untere Halbbild angezeigt. Es können maximal 4096 Farben auf dem Bildschirm dargestellt werden. Das wird möglich durch das HAM-Verfahren (Hold and Modify). Dabei kann der Farbwert des benachbarten Punktes jeweils nur um eine Grundfarbe abweichen (Veränderung der Rot-, Grün- und Blauintensität). Der Soundchip kontrolliert alle Eingabe-/Ausgabe-Operationen sowie die 4 getrennt steuerbaren Oszillatoren für den Sound. Der RAM des Amiga beträgt standardmäßig bei den verschiedenen Modellen zwischen 256 KByte (A1000) und 3 MByte (A2500) und kann bis 8,5 MByte ausgebaut werden. Jeder Amiga wird mit Maus und 3,5-Zoll-Diskettenlaufwerk ausgeliefert.

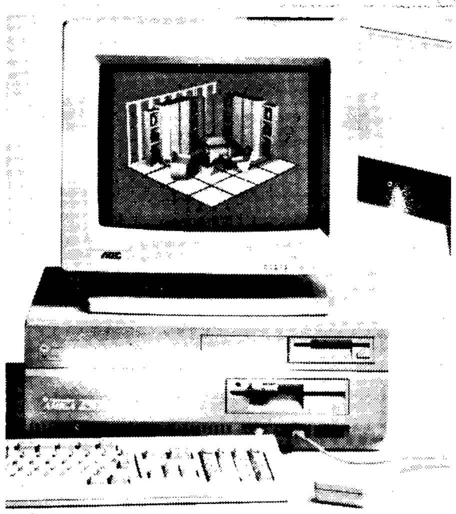
Die Software

Die grafische Benutzeroberfläche des Amiga (Workbench) ist multitaskingfähig, das heißt, es können mehrere Programme quasi gleichzeitig abgearbeitet werden, was aber zum Beispiel für aufwendige Grafikanwendungen (Videoscape und Modeler 3D von Aegis) erst effizient mit einem größeren Speicher und möglicherweise einer Turbokarte (MC 68020 und MC 68882) ist. Die Workbench ist nach stark ergonomisch motivierten Gesichtspunkten aufgebaut. Dort werden alle Programme durch ein Symbol (Icon) repräsen-

Die Amiga-Familie im Vergleich

	Amiga 1000	Amiga 500	Amiga 2000	Amiga 2500/20	Amiga 3000
Erscheinungsjahr	1985	1987	1987	1989	1990
RAM	256 KByte	512 KByte	1 MByte	3 MByte	5 MByte
CPU Koprozessor	MC 68000 /	MC 68000 /	MC 68000 /	MC 68020/30 MC 68881 oder MC 68882	MC 68030 MC 68882
Taktfrequenz	7,14 MHz	7,14 MHz	7,14 MHz	14,2 MHz oder 25 MHz	30 MHz
Laufwerke	1 × 3,5-Zoll-Floppy	1 × 3,5-Zoll-Floppy	1 × 3,5-Zoll-Floppy	1 × 3,5-Zoll-Floppy, 40-MByte-HD	1 × 3,5-Zoll-Floppy, HD
Betriebssystem, ROM	kein ROM (Diskette)	Kickstart 1,2	Kickstart 1,2	Kickstart 1,3	Kickstart 1,4
PC-Kompatibilität über	Sidecar	PC-Board	XT- oder AT-Karte	XT- oder AT-Karte	XT- oder AT-Karte

tiert. Standardmäßig sind auf der Workbench mehrere Programme zugänglich, unter anderem eine Textverarbeitung (Notepad), ein Taschenrechner, eine Uhr, ein Sprachausgabemodul, ein Kopierprogramm, ein Voreinsteller sowie das CLI (Command Line Interface) für die Kommunikation mit dem AmigaDos. Das A-DOS erinnert in seiner Funktion und im Aufbau sehr dem MS-DOS der Standard-PCs. Alle genannten Programme werden in separaten Fenstern abgearbeitet, wovon mehrere gleichzeitig geöffnet werden können. Dabei kann die Fenstergröße, der Fensterausschnitt und die Lage auf dem Bildschirm mit der Maus variiert werden. In vielen Amigaprogrammen wird mit Pulldown-Menüs (nach unten abrollende Menüs) gearbeitet, auch diese können mit der Maus angesprochen werden. Die Maus spielt überhaupt eine große Rolle beim Umgang mit dem Amiga. Mit diesen „Cursortasten der Zukunft" werden fast alle Operationen in den Programmen gesteuert. Für die Maus wie für das Workbenchkonzept sprechen ihre hohe Ergonomie. Eine Einarbeitung in die Workbenchoberfläche ist auch von jedem Anfänger in kürzester Zeit möglich. Auch alle anderen für den Amiga verfügbaren Programme sind meist in einer mausgesteuerten Fensteroberfläche zugänglich. Anwenderprogramme sind für viele Einsatzgebiete zu haben, angefangen bei Textverarbeitungen (Vizawrite) über Datenbankprogramme wie Superbase (auch kombinierte Bild- und Textausgabe) und Animations- und Grafiksoftware, wo sogar 3D-Animation möglich wird. DTP, Sound- oder Midianwendungen sind ebenfalls vorhanden. Es werden auch Compiler und Interpreter für die verschiedensten Sprachen angeboten wie Basic, Pascal, C, Assembler u. a.
Ein weiteres Programm eröffnet den Weg zur „größten Softwarebibliothek der Welt". Der MS-DOS-Transformer, mit ihm ist ein Großteil der DOS-Software auch für den Amiga verfügbar. Die Einschränkungen liegen in der Verminderung der Geschwindigkeit auf zirka 20 Prozent und in der fehlenden Grafikmöglichkeit. Unter dem Transformer wurden unter anderem die Programme dBase III Plus und Supercalc 4.0 getestet, sie wurden relativ gut abgearbeitet. Seit kurzem wird für den Amiga auch ein Mac-Emulator angeboten, und wenn man bereit ist, sich dazu ein Original-Mac-Laufwerk anzuschaffen, eröffnet



Die neue Version des Amiga 2500/30 von Commodore arbeitet mit dem 25-MHz-Motorola-Prozessor 68030, dem gleichschnellen mathematischen Koprozessor 68882 und den Amiga-spezifischen drei Koprozessoren für Datenverwaltung, Video, Grafik und Ton. Der 2500/30 ist durch den auf 32 Bit verdoppelten Datenbus und das integrierte Memory Management mehr als doppelt so schnell wie das bisherige Spitzenmodell mit dem 68020er-Prozessor.

sich hier ein weiteres interessantes Anwendungsfeld.

Erweiterungsmöglichkeiten

Der Amiga 2000 kann wegen seiner offenen Systemarchitektur (OSA) je nach Anforderungsniveau optimal ausgebaut werden. Es können z. B. Brückenkarten mit i8088 oder i80286 eingesetzt werden. Damit kann man aus einem Amiga einen vollwertigen PC/XT oder /AT machen.
In Kürze soll der A2500UX erscheinen, dieser ist mit einem MC 68030 ausgerüstet und durch das Betriebssystem Amix voll Unix-kompatibel, es stellt eine Portierung des Unix System V Release 3.1 von AT&T dar. Weiterhin sind Speichererweiterungskarten, Grafik- und Prozessorkarten einsetzbar. Selbst der kleine Amiga 500 ist intern auf 2,3 MByte aufrüstbar und mit einer MC 68020-Turbokarte und einem MC 68881-Koprozessor ist er am unteren Ende der Workstationklasse angelangt.
Peripher ist jeder Amiga mit Festplatten- oder Wechselplattenlaufwerken und 5,25- bzw. 3,5-Zoll-Floppy-Laufwerken sowie Video- und Soundsystemen erweiterbar – beim Amiga 2000 auch in Form von Gehäuseeinschüben. Im Bereich Video wird mit den Grafikeigenschaften des Amiga sogar eine neue Form des Computereinsatzes definiert: DTV = Desktop Video. Dieses Konzept basiert auf der Einbindung von Computergrafiksequenzen in Videofilme über das Genlockinterface. Damit ist eine Mischung der Video- und Computersignale stufenlos möglich. Mit einem Videodigitizer können auch Bilder aufgenommen und verfremdet werden, teilweise ist das sogar in Echtzeit möglich. Der Einsatzbereich auf diesem Gebiet reicht mit der entsprechenden Ausstattung bis zur Studioqualität. Der Amiga findet teilweise auch schon bei Fernsehproduktionen Anwendung.

Praktische Anwendung

Dem Amiga wurde lange Zeit nachgesagt, daß er nur als Heimcomputer zu gebrauchen sei. Doch das ist keineswegs der Fall. Er ist dank der kurzen Einarbeitungszeit in das System gut für alle Büroanwendungen, wie Textverarbeitung, Dateiverwaltung, Tabellenkalkulation und auch DTP, einsetzbar. Aber auch spezialisierte Einsatzgebiete sind denkbar, beispielsweise die Computersimulation in allen Bereichen. Ein interessanter Anwendungsfall ist auch der Einsatz eines Amiga in der Universitätsklinik Wien, wo ein System zur Planung von chirurgischen Eingriffen im Gehirn erarbeitet wurde. Auch als Informationssystem in öffentlichen Einrichtungen oder auf Messen ist der Amiga im Einsatz.

Ausblick

Der Amiga wird sicher noch weitere Zeichen in Richtung Grafik und Animation setzen. Er kann als Computer mit großer Leistung und geringem Preis in Zukunft vermutlich noch größere Marktanteile erringen. Nicht zuletzt wird der Amiga 500 dazu einen großen Anteil leisten. Der neue Amiga 3000 soll als perfektes Profiwerkzeug aber auch einen festen Kundenkreis finden. Zu denken wäre besonders an Werbeagenturen und Videostudios oder weitere anspruchsvolle Anwender.

TERMINE

19. Jahrestagung „Grundlagen der Modellierung und Simulationstechnik"

WER? Kammer der Technik, Wissenschaftlich-technische Gesellschaft für Meß- und Automatisierungstechnik

WANN? 18. bis 20. Dezember 1990

WO? Rostock

WAS?
- Grundlagen der Simulation, Applikation
- Simulation räumlich verteilter Systeme
- Simulation für den Schaltkreisentwurf
- Modellierung und Simulation in der Wasserwirtschaft
- Simulation und neuronale Netze

WIE? Interessenten wenden sich bitte an: Kammer der Technik, WGMA, Clara-Zetkin-Straße 115/117, Berlin, 1086; Tel. 2265218

Müller

Jahrestagung der WGMA „Flexible Automatisierung für die Fertigung"

WER? Kammer der Technik, Wissenschaftlich-technische Gesellschaft für Meß- und Automatisierungstechnik

WANN? 5. bis 6. Dezember 1990

WO? Erfurt

WAS?
- Stand und Perspektive bei Komponenten und Systemen unter dem Aspekt des Informationsflusses
- CNC-, SPS-, Roboter- und Transportsteuerungen
- Leitrechner, Leitstandstechnik
- Strukturfragen bei CIM-Lösungen zur Gewährleistung von Flexibilität, Qualitätssicherung

WIE? Interessenten wenden sich bitte an: Kammer der Technik, WGMA, Clara-Zetkin-Straße 115/117, Berlin, 1086; Tel. 2265218

Müller

Tagung „Speicherprogrammierbare Steuerungen"

WER? Kammer der Technik, Wissenschaftlich-technische Gesellschaft für Meß- und Automatisierungstechnik

WANN? 4. bis 6. Juli 1990

WO? Leipzig

WIE? Interessenten wenden sich bitte an: Kammer der Technik, WGMA, Clara-Zetkin-Straße 115/117, Berlin, 1086; Tel. 2265304

Bahn

Einführung in MS-DOS

Teil 2

Wolfram Schulze, Uwe Schulze, Berlin

Nachdem wir in der ersten Folge (in MP 4/90) den Systemstart und die Arbeit im hierarchischen Dateisystem behandelt haben, steht diesmal das Anpassen (Konfigurieren) von DOS an die eigenen Anforderungen im Vordergrund. Außerdem befassen wir uns mit Aufbau, Formatierung und Prüfung von Disketten. Weiter im Programm: Die Veränderung der DOS-Bereitschaftsanzeige (Prompt).

Wir richten uns ein

Interne und externe Kommandos

Der erste Teil des Kurses schloß mit der Vorstellung einiger wichtiger Kommandos. DOS unterscheidet dabei zwischen internen und externen Kommandos. Als intern werden alle diejenigen Befehle bezeichnet, die vom Kommandointerpreter COMMAND.COM ausgeführt werden können. Es bedarf keiner zusätzlichen Programme auf der Diskette, denn sie befinden sich stets im Speicher. Alle bisher besprochenen Befehle gehörten dieser Gruppe an. Soll dagegen ein externes Kommando ausgeführt werden, so kann es schon einmal zu folgender Ausschrift kommen:

Falscher Befehl oder Dateiname

Nämlich dann, wenn die entsprechende Programmdatei nicht auf der Diskette (oder Festplatte) vorhanden ist, denn jedes externe Kommando steckt in einer eigenen Datei vom Typ COM (selten EXE).

Warum diese Einteilung in interne und externe Kommandos? Nun, die wichtigsten und am häufigsten verwendeten Befehle sollen stets zur Verfügung stehen, andererseits würden aber komplexe Kommandos zu viel Platz im Speicher einnehmen – zumal, wenn sie nur selten benötigt werden. Die Anzahl externer Kommandos hat sich von Version zu Version vergrößert, bzw. die alten wurden erweitert. Oftmals hilft also nur ein Blick ins Handbuch um festzustellen, ob ein bestimmter Befehl bei dieser Version bereits unterstützt wird. Auch ist Vorsicht beim Austausch der DOS-Systemkommandos geboten. Die Mehrzahl läuft nur unter der dazugehörigen Betriebssystemsversion. Das gilt auch für den Kommandointerpreter. Viele DOS-Kommandos haben durch das Auftauchen komfortabler Dienstprogramme ihre Bedeutung verloren, manche müssen sich der Konkurrenz modernerer Ausführungen erwehren (z. B. der Debugger DEBUG) und eines gehört mit Sicherheit ins Museum: der Kommandozeileneditor EDLIN.

Zwei wertvolle Kommandos wollen wir uns näher ansehen: **XCOPY** ist ein erweiterter Kopierbefehl, der ab DOS-Version 3.20 zur Verfügung steht. Die Ausgabe von Quell- und Zieldateien erfolgt wie bei COPY.

Beispiele:

xcopy *.txt a: /p Kopieren aller Textdateien aus dem aktuellen Verzeichnis nach A: mit Abfrage

xcopy *.* a: /s Kopieren aller Dateien und Unterverzeichnisse nach A:

Zu beachten ist, daß beim letzten Beispiel nicht notwendigerweise die gesamte Diskette kopiert wird. Befindet man sich schon in einem Unterverzeichnis, so werden Dateien und Verzeichnisse nur ab dieser Hierarchiestufe kopiert. Das Doppeln von Disketten kann mit

xcopy a: b: /s

erreicht werden. DOS stellt dafür aber eine bessere Möglichkeit zur Verfügung: Den Befehl **DISKCOPY**. Mit

diskcopy a: b:

können Sie ein Duplikat der in A: befindlichen Diskette in B: anlegen. Kopiert wird physisch (Spur für Spur), deshalb ist dieser Befehl schneller als XCOPY. Die Zieldiskette braucht nicht formatiert zu sein; sie erhält automatisch das Quellformat. Aber Achtung: Alle Daten der Zieldiskette werden irreparabel überschrieben. Hüten Sie sich davor, Quell- und Ziellaufwerk zu vertauschen!

Der Befehl gestattet es auch, auf Rechnern mit nur einem Laufwerk Kopien zu fertigen. Mit

diskcopy a:

dient A: als Quell- und Ziellaufwerk. In manchen Versionen muß es auch doppelt angegeben werden:

diskcopy a: a:

Nachdem DOS so viel wie möglich in den Speicher geladen hat, fordert es zum Diskettenwechsel auf, beschreibt die Zieldiskette usw. Ein weiterer Vorteil des DISKCOPY-Befehls besteht darin, daß der Diskettenname (Volumenname – Sie erinnern sich?) mitkopiert wird. Das kann bei Sicherheitskopien von gekaufter Software wichtig sein (und wird in den Handbüchern empfohlen), nämlich dann, wenn die Programme diesen Namen abfragen.

Es soll nicht verschwiegen werden, wann Sie DISKCOPY nicht verwenden können: Zwischen Disketten verschiedener Formate kann nicht kopiert werden. Und soll die starke Fragmentierung (Zersplitterung von Programmteilen) auf einer Diskette aufgehoben werden, so muß, wie oben gezeigt, auf XCOPY zurückgegriffen werden.

An dieser Stelle sei auch auf die Befehle **BACKUP** und **RESTORE** verwiesen, die für Sicherungskopien der Festplatte vorgesehen sind und mit denen es möglich ist, größere Dateien zu kopieren als der Zieldatenträger aufnehmen kann. Beachten Sie dabei bitte, daß beide Befehle unbedingt zur gleichen Version von DOS gehören sollten; ohne ein passendes RESTORE gibt es keine Möglichkeit, mit BACKUP gesicherte Daten je wieder zugänglich zu machen.

Anlegen neuer Disketten

Die gespeicherten Informationen sind auf den Disketten in Spuren abgelegt, die wiederum in Sektoren unterteilt sind. Die Sektoren sind bei MS-DOS einheitlich 512 Byte groß (während bei CP/M recht verschiedene Sektorgrößen Verwendung fanden: Zuerst 26 × 128 Byte, später 16 × 256 und 5 × 1024 Byte – ausführlich beschrieben in MP 2/1989). Auf Standarddisketten befinden sich gewöhnlich 9 Sektoren auf einer Spur, so daß das Format 9 × 512 Byte zustande kommt. Auch heute wird noch viel Software in diesem Format ausgeliefert (z. B. Büchern beiliegende Disketten). In der DDR findet überwiegend das zweite Format (720 KByte) Verwendung. Das *zweiseitige* Format bedeutet, daß das Laufwerk über zwei Schreib-/Leseköpfe verfügt und beide Seiten gleichermaßen genutzt werden. Noch mannigfaltiger ist das Angebot an Festplatten. Als Zylinder werden die jeweils übereinanderliegenden Spuren des Plattenstapels bezeichnet.

Vor der erstmaligen Benutzung eines Datenträgers muß dieser formatiert (oder auch initialisiert) werden. Das geschieht heute ausschließlich softwaremäßig (man spricht deshalb auch von Softsektorierung). Damit wird erst beim Formatiervorgang festgelegt, mit welchem Format ein Datenträger verwendet werden soll; durch erneutes Formatieren (allerdings bei Verlust aller Daten) kann das auch geändert werden. Der **FORMAT**-Befehl (ein externer Befehl) wird wie folgt verwendet:

format a:

zum Formatieren einer Diskette in Laufwerk A:. Soll eine Systemdiskette eingerichtet werden, so kann das mit

format a: /s

geschehen. Ohne weitere Schalter wird stets das Standardformat des jeweiligen Laufwerks angenommen. Ein beliebter Fehler bei der Verwendung dieses Befehls liegt in der falschen Laufwerksangabe. Also Vorsicht: Alle Daten werden zerstört. Am besten, sie entfernen die Disketten aus den anderen Laufwerken.

```
Diskette duplizieren

DISKCOPY lw1: lw2: [/1]

lw1: und lw2:    Laufwerksangaben
/1               nur einseitig
```

```
Erweiterter Kopierbefehl

XCOPY [lw:] [pfad] [datei] [lw:] [pfad] [datei]
      [/a] [/d:datum] [/e] [/m] [/p] [/s] [/v] [/w]

/a         nur Dateien, die seit dem letzten Backup
           geändert wurden
/e         auch leere Subdirectories werden kopiert
           (sofern /s gesetzt ist)
/m         wie /a, Archivbit wird zurückgesetzt
/s         Subdirectories werden mitkopiert
/v         Vergleichslesen wird eingeschaltet
/w         warten auf Tastendruck zum Disketten-
           wechsel
/d:datum   nur Dateien; die nach dem Datum ge-
           ändert wurden, werden kopiert
```

Beim Formatieren wird der Datenträger durch ein Kontrollesen geprüft, um so Materialfehler festzustellen.
Der Formatbefehl kann auch für Festplatten genutzt werden, allerdings sind vorher zwei weitere Schritte nötig: Die Vorformatierung mit einem speziellen Programm oder einem Sprung ins BIOS und die Aufteilung in Bereiche (Partitions) durch **FDISK**. Diese Arbeit sollte aber einem Experten überlassen werden. Auf jeden Fall kann der Inhalt einer Festplatte dadurch nicht versehentlich mit FORMAT zerstört werden.
Ist nicht mit dem Schalter /s formatiert worden, so können auch noch nachträglich mit
sys a:
die Systemdateien BIO.COM und DOS.COM übertragen werden. Der COPY-Befehl ist dafür nicht geeignet, da diese Dateien in einem festgelegten Bereich der Diskette stehen. Deshalb muß der Datenträger auch leer sein oder dieser Platz beim Formatieren mit dem /b-Schalter reserviert worden sein, da sonst wichtige Bereiche überschrieben werden. Anschließend ist noch COMMAND.COM mit einem beliebigen Kopierprogramm zu übertragen.
Jedem Datenträger kann ein Name zugeordnet werden. DOS stellt dafür den **LABEL**-Befehl zur Verfügung. Label bedeutet bekanntlich soviel wie Marke oder Etikett. Im EDV-Sprachgebrauch wird auch der Name von Datenträgern selbst als Label bezeichnet. Mit
label a: pit
erhält der Datenträger in L ıufwerk A: den Namen PIT. Alle Kleinbuchstaben werden automatisch in Großbuchstaben umgewandelt. Der Name kann bis zu 11 Zeichen lang sein und – im Gegensatz zu Datei- und Verzeichnisnamen – Leerzeichen enthalten. Ausgeschlossen sind alle Zeichen, die auch für Dateinamen nicht verwendet werden dürfen.
Die Eingabe von
label
ohne einen Namen erzeugt folgende Ausschrift:

Diskette/Platte, Laufwerk A: hat keinen Namen
Name (max. 11 Zeichen) oder Eingabetaste
Besitzt der Datenträger bereits einen Namen, so wird dem Nutzer außerdem angeboten, diesen zu löschen. Der LABEL-Befehl kann jederzeit verwendet werden, also nicht nur solange der Datenträger leer ist. SYS und LABEL sind externe Befehle. Zur Erinnerung sei auch noch einmal auf den VOL-Befehl zum Anzeigen des so vergebenen Namens hingewiesen und, wie Ihnen sicher schon aufgefallen ist, wird er auch beim DIR-Befehl stets mit ausgegeben.

Überprüfen von Datenträgern

Ist ein Datenträger längere Zeit im Gebrauch, so können sich Inkonsistenzen (logische Widersprüche) im Dateisystem einstellen. Damit sind nicht mögliche Materialfehler gemeint – sie würden bereits beim Formatieren aufgespürt werden. Vielmehr handelt es sich um Fehler, die durch Stromausfall, Systemabsturz oder fehlerhafte Programme verursacht werden und die bewirken, daß Dateien nicht korrekt abgeschlossen sind. Die Folge ist, daß als belegt markierte Bereiche der Diskette (sie werden Cluster genannt) zu keiner Datei mehr gehören, also verloren sind, oder daß Bereiche mehreren Dateien zugeordnet sind, was den Inhalt verfälscht. DOS stellt das Programm **CHKDSK** zur Verfügung, das eine Prüfung von Datenträgern (Check Disk) vornimmt. Dazu werden die Verzeichniseinträge (Directories) herangezogen, sowie eine Tabelle, die auf jedem Datenträger vorhanden ist und die Belegung aller Cluster enthält. Diese heißt FAT (File Allocation Table) und ist im Augenblick für uns nicht weiter von Bedeutung.
chkdsk c:
überprüft den Datenträger des Laufwerks C: (i. a. eine Festplatte) und erzeugt folgendes Protokoll:

Diskette/Platte MO & PIT
erstellt 1 Jan 1980, 4.42
21309440 Byte Gesamtkapazität
81920 Byte in 6 geschuetzten Dateien
110592 Byte in 46 Verzeichnis(sen)
21053440 Byte in 1182 Benutzerdatei(en)
6144 Byte in fehlerhaften Sektoren
57344 Byte auf Diskette/Platte verfügbar

655360 Byte Gesamtspeicher
410336 Byte frei

Der obere Abschnitt bezieht sich auf die Festplatte (20 MByte) und der untere auf den Hauptspeicher (640 KByte). Zu den geschützten Dateien gehören die Systemdateien BIO.COM und DOS.COM. Wie bereits im Teil 1 beschrieben, besitzen diese die Attribute von geschützten und versteckten Dateien. Deshalb werden sie auch beim DIR nicht angezeigt. Zu den Dateiattributen mehr in einer späteren Folge.
Die Statistik zeigt uns ferner die Anzahl der insgesamt vorhandenen Unterverzeichnisse (46) und Dateien (1182) sowie den davon eingenommenen Platz. Auf der Festplatte sind nur noch 57344 Byte frei; wir werden uns also bald von einigen Daten trennen müssen.
Um eine Vorstellung von den Datenmengen zu vermitteln, werden als Faustregel für eine Seite Text 2 KByte veranschlagt. Mehr als 30 Seiten wären also auf der Festplatte nicht mehr unterzubringen.
Bei den hier angegebenen fehlerhaften Sektoren handelt es sich um Bereiche, die bereits beim Formatieren ausgemustert wurden. Die Platte wäre in diesem Falle also recht voll, aber in Ordnung. Kritisch sind Fehler, wenn CHKDSK feststellt, daß Cluster zu keiner Datei gehören oder doppelt zugeordnet sind. In diesem Falle sollte der /f-Schalter angegeben werden; nach Rückfrage versucht CHKDSK diesen Fehler zu beheben. In den meisten Fällen ist eine Reparatur möglich, wenn nicht, muß darüber nachgedacht werden, alle Dateien zu retten und den Datenträger neu zu formatieren.
CHKDSK kann auch dazu benutzt werden festzustellen, wie stark einzelne Dateien fragmentiert sind. Die Verwaltung der Dateien führt im Laufe der Zeit dazu, daß sie nicht mehr zusammenhängend abgespeichert sind. Sie werden „zerstückelt" oder – fragmentiert. Die Folge sind längere Ladezeiten, verursacht durch das Zusammensuchen der einzelnen Teile auf dem Datenträger. Auskunft darüber gibt CHKDSK mit Angabe eines Filenamens, also
chkdsk turbo.exe
zum Feststellen der Fragmentierung des Turbo-Pascal-Compilers.
Selbstverständlich können auch die Joker für mehrere Dateien Verwendung finden. Ein Aufheben der Fragmentierung kann nur durch Umkopieren erreicht werden (siehe Diskussion zu XCOPY) oder durch spezielle Verdichtungsprogramme (z. B. COMPRESS).

AUTOEXEC.BAT

Stets wiederkehrende Befehlsfolgen können im DOS in speziellen Kommandodateien zusammengefaßt werden. Sie tragen den Typ BAT (von Batch) und werden deshalb auch Batch-Dateien oder Stapeldateien (aus dem Großrechner-Vokabular) genannt. An einem

```
Formatieren von Datenträgern

FORMAT lw: [/1] [/4] [/8] [/s] [/v] [/b]

lw:     Laufwerk
/1      formatiert nur einseitig
/4      formatiert 360 KByte-Disketten auf
        HD-Laufwerk
/8      formatiert 8 Sektoren/Spur (Standard: 9)
/s      die drei Systemdateien werden automatisch
        mit übertragen
/v      nach dem Formatieren wird nach einem
        Diskettennamen (Volumelabel) gefragt
/b      der Platz für das Betriebssystem wird
        reserviert, auch wenn inzwischen Dateien
        auf die Diskette kopiert werden
```

```
Übertragen des Betriebssystems

SYS lw:

Es werden nur die beiden Dateien des BIOS und BDOS
übertragen; für COMMAND.COM muß ein normaler Kopier-
befehl verwendet werden. Voraussetzung ist, daß der
Datenträger leer ist oder mit dem /b - Schalter for-
matiert wurde.
```

```
Namen für einen Datenträger vergeben

LABEL [lw:] [name]

lw:    Laufwerk
name   Zeichenkette (bis zu 11 Zeichen)
```

```
Überprüfen eines Datenträgers

CHKDSK [lw:] [pfad] [datei] [/f] [/v]

lw:     Laufwerk
pfad    Zugriffspfad
datei   Dateiname
/f      nach Rückfrage ist eine Fehlerkorrektur
        möglich
/v      protokolliert, an welcher Datei jeweils
        gerade gearbeitet wird
```

DDR-SPEZIAL-ANGEBOT

Jetzt können Sie die Zeitschriften von Markt&Technik günstig abonnieren - in Mark und 1:1

DAS ABO

1/4 Jahr in Mark zum Kurs von 1:1 bis zur Währungsunion!

PC Magazin Plus: die Spezialistenzeitschrift für Computertechnik und Programmierung 3 Ausgaben inklusive Super-Utility-Diskette

Computer live: das Populäre Magazin für Kaufberatung 3 Ausgaben inklusive Super-Utility-Diskette

Computer Persönlich: Für den PC-Anwender im Beruf 6 Ausgaben inklusive Super-Utility-Diskette

64er: Das Magazin für Computerfans 3 Ausgaben inklusive Startdiskette

INKLUSIVE BEGRÜSSUNGS-DISKETTE!

BESTELLCOUPON

Schicken Sie den ausgefüllten Coupon an: IVE Info-Verlag, Im Applikationszentrum, Elektronik Berlin, Mainzerstraße 25, Berlin 1035

Ja, ich möchte ihr Spezialangebot nutzen und bestelle folgende Zeitschrift(en) für 1/4 Jahr inklusive Begrüßungsdiskette:

- ☐ 64er - das Magazin für Computerfans (1/4 Jahresabo inkl. Startdiskette: 3 Ausgaben für 20,-M)
- ☐ Computer live - das Internationale Computer Magazin (1/4 Jahresabo inkl. Super-Utility-Diskette: 3 Ausgaben für 18,- M)
- ☐ PC Magazin PLUS - die Spezialistenzeitschrift für Computertechnik (1/4 Jahresabo inkl. Super-Utility-Diskette: 3 Ausgaben für 21,-M)
- ☐ Computer Persönlich - für den PC-Anwender im Beruf (1/4 Jahresabo inkl. Super-Utility-Diskette: 6 Ausgaben für 34,-M)

Das Abonnement läuft 1/4 Jahr und verlängert sich automatisch um ein weiteres 1/4 Jahr bis zur Währungsunion. Ich kann jederzeit zum Ende des bezahlten Zeitraumes kündigen.

Name, Vorname Straße, Hausnummer

PLZ, Wohnort Datum, Unterschrift

02/ AC 72 05

Einladung zur 1. Vollversammlung mit der Möglichkeit des Beitritts zum

UNTERNEHMENSVERBAND INFORMATIONSSYSTEME e.V.

UVI

am 27. Juni 1990 um 10.30 Uhr
im Zentralinstitut für Kybernetik und Informationsprozesse
der Akademie der Wissenschaften der DDR
(Kurstraße 33, Berlin 1086)

Im Unternehmensverband Informationssysteme (UVI) e.V. schließen sich auf dem Territorium der DDR tätige Unternehmen, Organisationen und Institutionen zusammen, die auf dem Gebiet der Informations- und Kommunikationssysteme Produkte und Dienstleistungen anbieten.

Weitere Informationen über die Geschäftsstelle des UVI:

Akademie der Wissenschaften der DDR
Institut für Theorie, Geschichte und Organisation der Wissenschaft
Herr Dr. sc. M. Weber
1100 Berlin, Prenzlauer Promenade 149-152,
Tel.: 4797 275

Mit Platinenfolien

ELV journal

1/90 Feb./März Fachmagazin für angewandte Elektronik 5,80 DM

- Video-Verteiler + Fernbedienungsverlängerung
- 12 V-Laser-Power-Supply
- 230 V-LED
- S-VHS/FBAS-Konverter
- Laser-Modulator-Interface
- 5 Labor-Netzgeräte: 0-15 V/4 A bis 0-60 V/1 A
- PC-Karten-Tester
- Breitband-Antennenverstärker
- 200 A-Präzisions-Leistungs-Shunt
- LED-Tester
- Prozessor-Frequenzzähler
- Leistungshalbierer

Frankreich srf 22,– / Schweiz chf 5,80 / Niederlande nlg 7,95 / Luxemburg 150,– / Belgien bec 150,– / Dänemark dkk 28,– / Schweden sek 25,– / Norwegen nok 27,– / Finnland fim 17,– / Österreich öS 57,–

ELV journal ... entdecken

Mit Platinenfolien

Mehr Ideen

Jedes Heft bietet Ihnen 10 -15 ausgereifte Schaltungen zum Selbstbau aus fast allen Bereichen der Elektronik. Von anspruchsvollen Meß- und Prüfgeräten, Umwelt- und Medizintechnik über ein vielseitiges Schulgeräteprogramm bis hin zu Video- und Audiotechnik, Kfz-Elektronik, Haustechnik, Computer- oder Laser- und Lichttechnik reicht das ELV-Produktionsprogramm. Das sind über 70 Geräte mit hervorragendem Gebrauchswert pro Jahr.

Mehr Qualität

Jedes Gerät wird von uns ausführlich getestet und hat einen hohen technischen Standard. Um die Funktionssicherheit zu erhöhen, werden selbstverständlich nur Qualitätsbauteile eingesetzt.

Mehr Sicherheit

Die Nachbausicherheit ist durch ausführliche Funktionsbeschreibungen und leicht verständliche Bauanleitungen garantiert. Die praktische Anwendung der Geräte wird durch Beispiele erläutert.

Mehr Service

Bei ELV erhalten Sie von der Idee bis zum fertigen Gerät alles aus einer Hand. Dadurch sind unsere Fachleute in der Lage, Sie bei allen auftauchenden Fragen schnell und unkompliziert zu beraten. Ein kurzer Brief genügt.

Mehr Leistung

Das ELVjournal erscheint 6x im Jahr
Am bequemsten für Sie im Abonnement.
Denn für nur 32,40 M im Jahr erhalten Sie alle 2 Monate die druckfrische Ausgabe direkt ins Haus.
Entdecken Sie mit dem ELVjournal die Welt der Elektronik.
Fordern Sie Ihr kostenloses Probeheft an.

Abbildung des Fertiggerätes

umfassende Bedienungshinweise

ausführliche Funktionsbeschreibung mit Anwendungsbeispielen

detailgenaue Schaltpläne aus dem Entwicklungslabor

Von der Idee zum Gerät

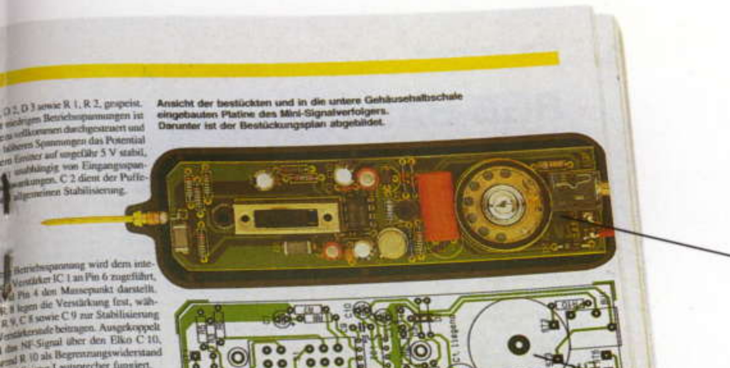
Unsere Ingenieure und Techniker entwickeln für Sie funktionssichere und absolut alltagstaugliche Qualitätsgeräte.
Durch eine intensive Zusammenarbeit mit Forschungsabteilungen der technischen Universitäten und Entwicklungsabteilungen der Industrie ist ein gleichbleibend hohes technisches Niveau garantiert.

High-Tech-Ausstattung

Modernste technische Ausrüstung wie CAD-Systeme und untereinander vernetzte Rechneranlagen unterstützen diese Fachleute bei ihrer täglichen Entwicklungsarbeit.
Die eigenständige Erstellung der Software für die benötigten Mikroprozessoren gehört ebenso dazu wie die Entwicklung von Anwenderprogrammen für die verschiedenen PC-Applikationen.

Mit dies
können
kostenl
des ELV
und einf
Am best
absende

Sie die Welt der Elektronik



..., D 2, D 3 sowie R 1, R 2, gespeist. ... niedrigen Betriebsspannungen ist ... vollkommen durchgesteuert und ... höheren Spannungen das Potential ... Emitter auf ungefähr 5 V stabil, ... unabhängig von Eingangsspan... ...wirkungen. C 2 dient der Puffe... ...allgemeinen Stabilisierung.

Ansicht der bestückten und in die untere Gehäusehalbschale eingebauten Platine des Mini-Signalverfolgers. Darunter ist der Bestückungsplan abgebildet.

... Betriebsspannung wird dem inte... ...Verstärker IC 1 an Pin 6 zugeführt, ... Pin 4 den Massepunkt darstellt. R 8 legen die Verstärkung fest, wäh... R 9, C 8 sowie C 9 zur Stabilisierung ...Verstärkerstufe beitragen. Ausgekoppelt ... das NF-Signal über den Elko C 10, ...rend R 10 als Begrenzungswiderstand ...den Miniatur-Lautsprecher fungiert. ...Eingangssignalanpassung wird über ...stufigen Abschwächer S 1 / R 4 - R 7 im ...reich von 0 bis 60 dB vorgenommen.

Stückliste: Mini-Signalverfolger

Widerstände	
1Ω	R 9
4,7Ω	R10
33Ω	R 8
56Ω	R 2
100Ω	R 7
1kΩ	R 6
10kΩ	R 1, R 5
100kΩ	R 3, R 4
Kondensatoren	
330pF	C 9
100nF, ker	C 5
100nF	C 6
220nF	C 8
10µF/25V	C 4
47µF/16V	C 3
100µF/16V	C 2, C 7, C 10
100µF/63V	C 1
Halbleiter	
TBA820	IC1
2N3019	T2
BC556	T1
ZPD, 5,6 V	D4
1N4001	D1
1N4148	D2, D3
Sonstiges	
Schiebeschalter, 4 x um, print	S 1
Klinkenbuchse, 3,5 mm, mono, print	BU 1
1 Mini-Lautsprecher	
3 Lötstifte, 1,3 mm	
2 Krokoklemmen	
30 cm isolierte Leitung, 2adrig	
30 mm Silberdraht	

Über die vergoldete Tastspitze wird das NF-Signal an den Platinenanschlußpunkt ST 1 eingekoppelt. Gerade bei Tastspitzen ist eine sichere Kontaktgabe von großer Wichtigkeit, damit Fehlmessungen und, wie hier im Audiobereich, unnötiges Knacken und Prasseln vermieden werden (welcher Elektroniker hat sich nicht schon einmal über mangelnde Kontaktgabe bei Prüfspitzen und Tastköpfen geärgert). Hier bietet die hochwertige vergoldete Tastspitze eine optimale Voraussetzung für sichere Kontaktierungen.

Über C 6 gelangt das NF-Signal auf den 4stufigen Abschwächer, der mit S 1 und Zusatzbeschaltung aufgebaut wurde. In der eingezeichneten Schalterstellung gelangt das Signal unabgeschwächt über R 4 auf den Verstärkereingang (Pin 3) des IC 1. In der rechts daneben gezeichneten Schalterstellung bildet R 4 mit R 5 einen Spannungsteiler, wodurch eine Abschwächung um 20 dB erfolgt (genaugenommen um 20,83 dB). In der nächsten Stellung erfolgt eine Abschwächung von 40 dB (R 4 / R 6) und in der ganz rechten Schalterstellung eine Abschwächung um 60 dB (R 4 / R 7).

Durch die nachgeschaltete Verstärkung des IC 1 können Signale im Bereich von wenigen mV bis hin zu 3 V unverzerrt bearbeitet werden, wobei selbst 230 V Wechselspannung dem Gerät kurzzeitig keinen Schaden zufügen.

Zum Nachbau

Anhand des Bestückungsplanes werden zunächst die niedrigen und anschließend die höheren Bauelemente auf die Platine gesetzt und auf der Leiterbahnseite verlötet. Der Miniatur-Lautsprecher wird mit 2 kurzen Silberdrahtstücken an die Platinenanschlußpunkte ST 2 und ST 3 angelötet. Die 2adrige Versorgungsspannungszuleitung wird an die beiden Lötstifte der Platinenanschlußpunkte ST 4 (positive Versorgungsspannung) und ST 5 (Masse) angesetzt. Ein ca. 10 mm langes Silberdrahtstück verbindet die vergoldete Tastspitze mit dem Platinenanschlußpunkt ST 1.

Nachdem die Bestückung nochmals sorgfältig überprüft wurde, kann ein erster Test vorgenommen werden.

Für den Einbau in das handliche Tastkopfgehäuse sind noch die beiden Aussparungen an der Geräterückseite (hintere Stirnseite) vorzunehmen. Die rechteckige Ausstanzung für den Schiebeschalter sowie die Bohrungen für den Lautsprecher sind bereits im Gehäuse vorhanden. An der Stelle, an welcher der Hals der 3,5 mm-Klinkenbuchse durch das Gehäuse tritt, sind sowohl im Unter- als auch im Oberteil halbkreisförmige Aussparungen im Durchmesser des Buchsenhalses einzubringen. Hierfür ist eine Rund- oder Schlüsselfeile besonders geeignet. Unmittelbar daneben wird im Gehäuseoberteil eine kleine Aussparung für die 2adrige Versorgungsspannungszuleitung eingebracht.

Nun kann die Platine in die untere Gehäusehalbschale eingelegt werden, wobei gleichzeitig auch die Tastspitze eingesetzt wird. Die genaue Position ist der Abbildung zu entnehmen. Alsdann wird das Gehäuseoberteil aufgesetzt und mit 2 Knippingschrauben von der Unterseite her fest verschraubt. Nun kann dieses nützliche Testgerät seiner Bestimmung zugeführt werden.

ELV journal 2/90

31

Abbildung der fertig-bestückten Platine in Originalgröße

detailgenauer Bestückungsplan zum richtigen Einsetzen der Bauteile

leicht verständliche Nachbauanleitung

komplette Auflistung aller zum Bau benötigten Bauteile

Heftpreis 1 zu 1

Schnell sein lohnt sich !!

Die ersten 1.111 Einsender erhalten diesen praktischen Solarrechner im Scheckkartenformat zusätzlich zu Ihrem aktuellen ELV journal gratis.

er Karte
Sie Ihr
oses Exemplar
journal schnell
ach abrufen.
ten noch heute
en.

Gratis-Anforderungs-Karte

☐ **Ja**, ich möchte das ELV journal kennenlernen. Bitte schicken Sie mir sofort die aktuelle Ausgabe - **natürlich kostenlos** - an meine unten aufgeführte Adresse. Sollte ich unter den ersten 1.111 Einsendern sein, erhalte ich zusätzlich den praktischen Solarrechner.

Bitte hier Ihre Adresse eintragen:

Name: ______________________

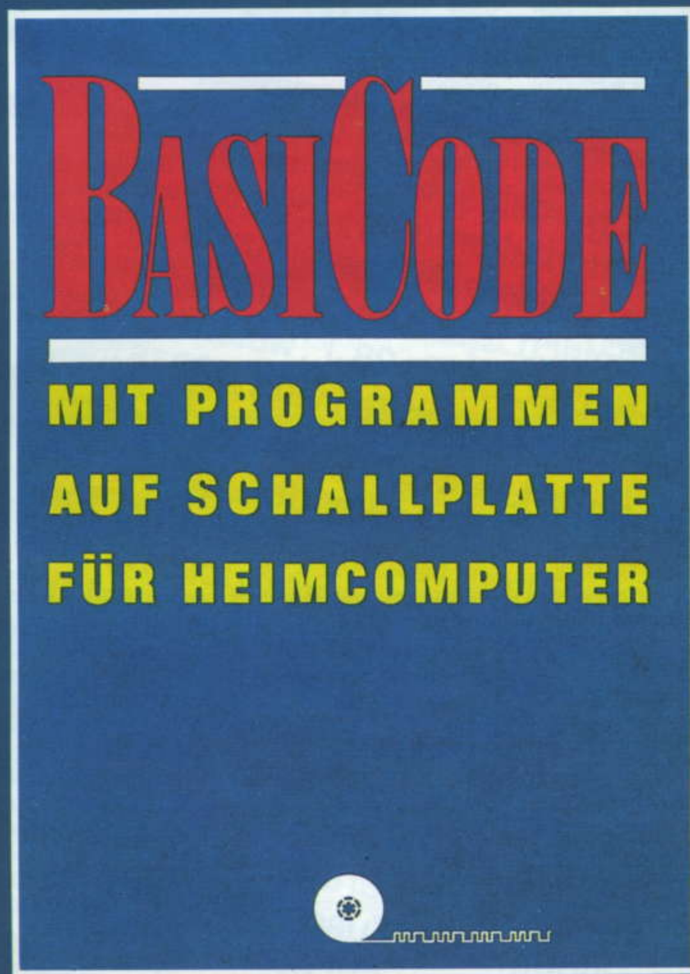
Vorname: ______________________

Straße: ______________________

PLZ: ________ Ort: ______________________

D3

Neu-erscheinung



VERLAG TECHNIK
BERLIN

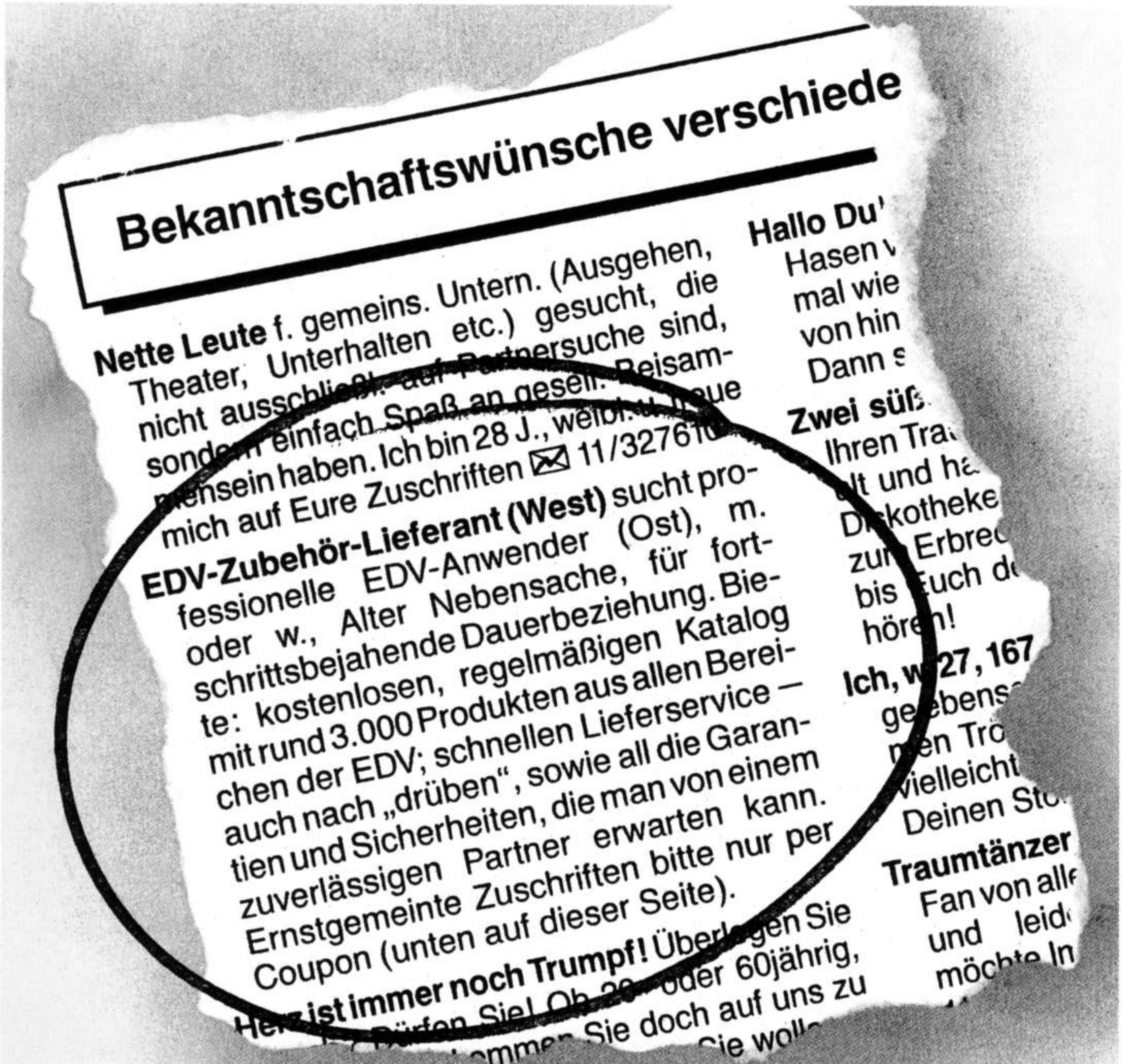
Auslieferung
in diesen Tagen
durch den Fachbuch-
handel!

Von Horst Völz
200 Seiten,
1 Datenschallplatte,
Broschur,
24,– M; 28,– DM.

Bestellangaben:
554 342 0/Völz, Basicode

BASICODE erfüllt die Wünsche vieler Computerfreunde:
Den direkten Austausch von Daten und Programmen zwischen verschiedenen Rechnern. Hierzu wurden ein einheitliches Datenformat und eine BASIC-ähnliche Sprache entwickelt. So ist es möglich, daß Texte, Grafik und Tonausgabe universell nutzbar sind.

BASICODE ist also das ideale Kommunikationssystem für Computerclubs, Arbeitsgemeinschaften, Schulen usw. Aber auch der „Stille Programmierer" wird viel Interessantes an BASICODE entdecken. Um die hohe Universalität erreichen zu können, benötigt jeder Rechnertyp ein spezielles Zusatzprogramm, den Bascoder. Für die wichtigsten Rechner befinden sich die Bascoder auf der zu diesem Buch gehörenden Schallplatte.



Bekanntschaftswünsche verschiede

Nette Leute f. gemeins. Untern. (Ausgehen, Theater, Unterhalten etc.) gesucht, die nicht ausschließl. auf Partnersuche sind, sondern einfach Spaß an gesell. Beisammensein haben. Ich bin 28 J., weibl. u. freue mich auf Eure Zuschriften ✉ 11/32761

EDV-Zubehör-Lieferant (West) sucht professionelle EDV-Anwender (Ost), m. oder w., Alter Nebensache, für fortschrittsbejahende Dauerbeziehung. Biete: kostenlosen, regelmäßigen Katalog mit rund 3.000 Produkten aus allen Bereichen der EDV; schnellen Lieferservice – auch nach „drüben", sowie all die Garantien und Sicherheiten, die man von einem zuverlässigen Partner erwarten kann. Ernstgemeinte Zuschriften bitte nur per Coupon (unten auf dieser Seite).

Herz ist immer noch Trumpf! Überlegen Sie ... oder 60jährig, Dürfen Sie! Ob 20 ... kommen Sie doch auf uns zu ... Sie woll...

Hallo Du' Hasen ... mal wie ... von hin ... Dann s...

Zwei süß... Ihren Tra... ...lt und ha... Diskotheke... zum Erbre... bis Euch d... hören!

Ich, w. 27, 167 ...gebens... ...en Trö... vielleicht ... Deinen St...

Traumtänzer Fan von all... und leid... möchte In...

Diese „Kontaktanzeige" ist kein Scherz – wir suchen tatsächlich. Denn wir wissen zwar, daß es in der DDR tausende von EDV-Anwendern gibt, die sicher gern einen Blick in unseren Katalog werfen würden. Aber wie kommt man zusammen? Wir haben uns für diesen Weg entschieden. Und wenn Sie sich von unserer Annonce angesprochen fühlen, würden wir uns freuen, von Ihnen zu hören. Wir – das ist Inmac, der größte unabhängige EDV-Zubehör-Lieferant und -Hersteller der Welt mit Niederlassungen in 9 Ländern. In der Bundesrepublik liegt unsere Zentrale in Raunheim bei Frankfurt. Und bis es uns auch in Leipzig, Dresden oder Rostock gibt, sind wir von hier aus für Sie da.

Schicken Sie einfach den Coupon dieser Anzeige an uns zurück und Sie erhalten sofort kostenlos den aktuellen Inmac-Katalog. In ihm finden Sie Disketten, EDV-Papier, Farbbänder, Kabel, Stecker, Diagnosegeräte, Pufferspeicher, Möbel und viel innovatives Zubehör – alles sofort lieferbar!
Dem Katalog beigelegt ist ein Info-Blatt mit Bestell-, Liefer- und Zahlungshinweisen für Interessenten aus der DDR. Auf gute Zusammenarbeit!

Datacom-Technik · EDV-Zubehör
inmac®

Bitte ausschneiden und an folgende Anschrift schicken:
Inmac GmbH, Frankfurter Straße 103, D-6096 Raunheim, z.Hd. Frau Dorschner

Ja, schicken Sie mir bitte kostenlos und unverbindlich den aktuellen Inmac-Katalog.

Name/Vorname ______________________

Firma ______________________

Abteilung ______________________

Straße/Nr. ______________________

PLZ/Ort ______________________

Telefon ______________________

Vier Antworten, die uns helfen, Ihre Bedarfslage einzuschätzen:

Zu welcher Branche gehört Ihr Betrieb?

____

Wieviele Beschäftigte arbeiten an dem genannten Firmenstandort?

☐ 1-49 ☐ 50-99 ☐ 100-499 ☐ 500-999 ☐ 1000 und mehr

Welche Art von EDV wird dort eingesetzt?

☐ Großrechner ☐ Mini-Computer ☐ Personal Computer

Marke(n) ______________________

Welches Gebiet der EDV ist für Sie besonders interessant?

____

kleinen Beispiel sei das einmal demonstriert. Wir benutzen als Editor den COPY-Befehl, wie in Teil 1 erläutert. Sie können aber auch Ihren Hauseditor benutzen. Folgende Kommandos dienen zum Festlegen eines kleinen Testfiles:

copy con test.bat
cls
ver
<CTRL> <Z>

Abgeschlossen wird mit der Tastenkombination <CTRL> <Z> (<Strg> <Z>). Die damit erzeugte Datei kann man sich mit

type test.bat

ansehen, und beim Aufruf mit dem Dateinamen

test

werden die enthaltenen Befehle automatisch ausgeführt. In diesem Falle wird der Bildschirm gelöscht und die DOS-Version angezeigt.

Auf diese Weise können beliebige Kommandos automatisch und wiederholt abgearbeitet werden. In der nächsten Folge des Kurses geben wir eine ausführliche Darstellung der Möglichkeiten der Batch-Programmierung sowie nützliche Tips und Tricks. An dieser Stelle soll nur eine ganz besondere Kommandodatei besprochen werden. Beim Systemstart sucht der Kommandointerpreter nach einer Datei mit dem Namen AUTOEXEC.BAT. Findet er sie nicht, so werden Datum und Zeit abgefragt, ist sie vorhanden, werden die enthaltenen Befehle abgearbeitet. Damit hat der Nutzer ein Mittel in der Hand, beim Start stets wiederkehrende Befehlsfolgen ausführen zu lassen. Eine typische AUTOEXEC.BAT (auch Start Up File genannt) könnte folgendes Aussehen haben:

echo off
break on
verify on
ver
prompt pg
path c:\dos; c:\tools
append c:\dos; c:\tools
keyb gr
date
time

Mit **echo off** wird die Protokollierfunktion (auch Echofunktion genannt) ausgeschaltet. Die folgenden Befehlszeilen erscheinen somit nicht mehr auf dem Bildschirm – wohl aber die durch sie erzeugten Ausgaben. Diese Anweisung dient nur der Erhöhung der Übersichtlichkeit. Am Ende der Batch-Datei schaltet sich das Echo automatisch wieder ein; wird der Befehl direkt von DOS aus gegeben, so muß es explizit mit **echo on** wieder eingeschaltet werden. Wird echo ohne Parameter eingegeben, zeigt der Bildschirm die gegenwärtige Schalterstellung an. Außerdem kann der Befehl echo dazu benutzt werden, um Nachrichten auf den Bildschirm auszugeben; eine Option, die nur in Stapeldateien einen Sinn gibt. Versuchen Sie es einmal mit

echo alles roger

Mit dem **BREAK**-Befehl (interner Befehl) wird festgelegt, wann die Tastenkombination <CTRL> <BREAK> (bei den meisten Rechnern auch <CTRL> <C>) Zum Programmabbruch führen soll. Mit break off kann die Überprüfung abgeschaltet werden (was auch Standard ist). Genaugenommen ist die Auswertung der Unterbrechungstasten nicht völlig unterbunden; bei einigen Ein- und Ausgabeoperationen sind sie trotzdem noch wirksam. Das Einschalten wird mit break on erreicht. Diese Option ist besonders für Programmierer geeignet, die eventuell vorkommende Endlosschleifen abbrechen müssen. Voraussetzung ist aber, daß DOS-Systemrufe enthalten sind, denn nur dort findet eine Auswertung der Tastenkombination <CTRL> <BREAK> (<Strg> <Untbr>) statt. Beispielsweise läßt sich so auch während der Diskettenarbeit ein Abbruch erzielen (z. B. während FORMAT).

Dem PROMPT-Befehl ist am Ende dieses Kurses ein ausführlicher Abschnitt gewidmet. Kommen wir deshalb gleich zum **PATH**-Befehl. Dieser wird in kaum einer AUTOEXEC.BAT fehlen. Er legt fest, wo DOS nach Programmen suchen soll, die sich nicht im aktuellen Verzeichnis befinden. Beispielsweise möchte man Wordstar oder ein anderes Textverarbeitungsprogramm seines Vertrauens von jedem Textverzeichnis aus aufrufen können, aber (aus Platzgründen) nicht in jedes Verzeichnis kopieren. Gleiches gilt für die externen DOS-Befehle. Wie Sie sich erinnern werden, handelt es sich dabei um eigene Programme. Wollen Sie also den XCOPY- oder den FORMAT-Befehl aufrufen, so muß DOS wissen, wo er zu finden ist – denn im aktuellen Verzeichnis ist er mit großer Wahrscheinlichkeit gerade nicht.

Zuerst wird stets im aktuellen Verzeichnis gesucht, dann der Reihe nach in den Verzeichnissen, die der PATH-Befehl angibt. Im Falle der hier als Beispiel gezeigten AUTOEXEC.BAT würden die Verzeichnisse DOS und TOOLS durchmustert werden. Erst wenn das Programm in allen diesen Pfaden nicht gefunden wird, erscheint

Falscher Befehl oder Dateiname

Es kann auch auf verschiedenen Laufwerken gesucht werden, beispielsweise in den Wurzelverzeichnissen von A: und C: mit

path c:\; a

Üblicherweise wird man aber nur auf der Festplatte oder der Systemdiskette suchen lassen, um nicht auf leere Laufwerke oder falsch eingelegte Disketten zuzugreifen. Mit

path

ohne Parameter wird der aktuelle Pfad angezeigt, und mit

path;

wird der Pfad gelöscht.

Der PATH-Befehl ist ein sehr wertvoller Befehl, es sei aber auch nicht verschwiegen, daß sich bei einem langen PATH die Ladezeiten verlängern. Nicht zu vergessen ist, daß mit PATH nur ein Suchpfad für ausführbare Programme (der Typen COM, EXE und BAT) festgelegt werden kann. Insbesondere mit nachladbaren Programmteilen (sogenannte Overlays), die nicht gefunden wurden, gibt es Probleme. Deshalb steht ab Version 3.20 der zusätzliche Befehl **APPEND** zur Verfügung. Im oben gezeigten Beispiel der AUTOEXEC.BAT sind beide Pfade identisch, es ist aber auch möglich, zwei verschiedene Suchpfade zu definieren, wenn Programmdateien und Nicht-Programmdateien in gesonderten Verzeichnissen stehen. Die Verwendung des APPEND-Befehls erfolgt analog dem PATH-Befehl. Ohne Parameter aufgerufen, zeigt er den aktuellen Suchpfad an, und mit

append;

wird der Pfad gelöscht. Es sei noch darauf verwiesen, daß PATH ein interner Befehl ist, während es sich bei APPEND um einen externen handelt.

Mit **KEYB GR** wird der deutsche Tastaturtreiber installiert; Standard ist die amerikanische Belegung. Sie werden das zuerst an der Lage des Doppelpunktes bemerken, der zum Umstellen der Laufwerke benötigt wird. Die amerikanische Tastatur kann mit der Tastenkombination <CTRL> <ALT> <F1> jederzeit wieder aktiviert werden, mit <CTRL> <ALT> <F2> gelangt man zurück zur deutschen Belegung. Mit einer anderen Landeskennung können auch weitere Tastaturtreiber anderer Länder geladen werden; nur die jeweils letzten beiden sind aber aktivierbar. Vor der DOS-Version 3.30 war die Handhabung für landessprachliche Tastaturtreiber etwas anders. Hier existierte für jedes Land ein eigener Treiber. Der entsprechende Befehl heißt

keybgr

Äußerlich besteht der Unterschied nur im fehlenden Leerzeichen, in Wirklichkeit existiert hier für jedes Land ein eigenes Kommando, während ab DOS 3.30 die Landeskennung nur ein Parameter für KEYB ist, der die jeweiligen Zeichensätze aus einer Datei holt.

Da die Datei AUTOEXEC.BAT eine ganz normale Batch-Datei ist, die sich nur durch ihren reservierten Namen auszeichnet, kann sie auch jederzeit durch Aufruf mit dem Namen

autoexec

gestartet werden. Umgekehrt können alle Befehle, die in ihr enthalten sind, auch einzeln auf Kommandoebene angewendet werden bzw. können beliebige andere DOS-Kommandos oder Programme in die AUTOEXEC.BAT geschrieben werden. Sollen beim Systemstart alle .BAK-Dateien gelöscht werden, so könnte das mit

del *.bak

in der AUTOEXEC.BAT geschehen. Beliebt ist die Möglichkeit, beim Systemstart gleich ein Programm aufzurufen. Wenn sie stets mit dBase arbeiten, so können sie als letzten Befehl

dbase

einfügen und gelangen beim Einschalten automatisch in das Anwenderprogramm. Zur Unterstützung unbedarfter Nutzer ist dies sehr hilfreich, da keine DOS-Kenntnisse nötig sind. Nach dem Studium unseres Kurses wird das für Sie aber überflüssig sein.

Die Konfigurationsdatei CONFIG.SYS

Beim Systemstart sucht DOS nach einer Datei, die Angaben zur Konfiguration enthält. Ihr Name ist CONFIG.SYS, und sie legt vor allem fest, welche Treiber (Gerätesteuerprogramme) geladen und welche Pufferbereiche reserviert werden sollen. Die Angaben in dieser Datei sind – im Gegensatz zu allem, was wir bisher besprochen haben – keine Befehle, die ausgeführt werden können. Deshalb werden sie im folgenden als Parameter bezeichnet. Eine typische CONFIG.SYS könnte folgendes Aussehen haben:

files = 20
buffers = 20

```
Protokollierfunktion
ECHO [on/off/text]

on      einschalten
off     abschalten
text    Ausgabe des Textes auf
        den Bildschirm
```

```
Prüfung auf Abbruch mit <CTRL> <BREAK>
BREAK [on/off]

on       Überprüfung bei jedem DOS-Ruf
off      Überprüfung nur bei Ein-/Ausgabe-
         Funktionen
```

```
Suchpfad für Programm-Dateien festlegen

PATH [=] [lw:] [pfad] [;[lw:][pfad]...]
bzw.
PATH;          zum Löschen des Pfades

=        keine Bedeutung, Benutzung wahlweise
lw:      Laufwerk
pfad     Suchpfad
```

```
Suchpfad für Nicht-Programm-Dateien

APPEND [lw:] [pfad] [;[lw:][pfad]...] [/x] [/e]
bzw.
APPEND;        zum Löschen des Pfades

lw:      Laufwerk
pfad     Suchpfad
/x       erweitert die Suche auch auf Programm-
         dateien
/e       speichert den Suchweg im Environment
         (wie bei Path)
```

```
Anpassung nationaler Tastaturen

KEYB [xx] [,nnn] [datei]

xx       Landeskennung: GR für Germany
nnn      Nummer des Zeichensatzes
datei    Name der Datei, die Tastatur-
         Spezifikation enthält
```

```
K o n f i g u r a t i o n s b e f e h l e
              in CONFIG.SYS

BREAK = ON        Prüfung auf Abbruch
        OFF       mit <CTRL> <BREAK>
                  ein-/ausschalten
                  (Standard: OFF)

BUFFERS = n       Anzahl der Puffer für Fest-
                  plattenzugriffe (1..99)

COUNTRY = n       Einstellung landesspezifischer
                  Schreibweisen (Datum, Zeit)
                  n  =  001   USA  (Standard)
                  n  =  049   Bundesrepublik
                              Deutschland
                  (entspricht der Vorwahl im
                   internationalen Telephon-
                   verkehr)

DEVICE = datei    Installation von Geräte-
                  treibern

FILES = n         maximale Anzahl gleichzeitig
                  offener Dateien  (8..255)

FCBS = m,n        maximale Anzahl gleichzeitig
                  offener File Control Blocks
                  (1..255) (CP/M-kompatibel)
                  m  Anzahl
                  n  Anzahl von FCBs, die nicht
                     geschlossen werden sollen,
                     wenn m überschritten ist

SHELL = [lw:] [pfad] [datei]        oder
SHELL = [lw:] [pfad] command.com [parameter]

                  Festlegung des Kommandointer-
                  preters
                  Entweder wird ein eigener
                  Kommandointerpreter installiert
                  oder COMMAND.COM mit Parametern
                  (siehe Beschreibung COMMAND.COM)
```

country = 049
device = \dos\ansi.sys
lastdrive = f

Mit **FILES** kann die Anzahl der gleichzeitig geöffneten Dateien festgelegt werden. Standard sind 8, höchstens können 255 angegeben werden. Zu beachten ist, daß in dieser Zahl auch die 5 Standard-Eingabe- und -Ausgabekanäle (Konsole, Drucker, Fehlerausgabe) enthalten sind, so daß bei einer Zahl von 8 nur noch drei nutzereigene Dateien verbleiben. Die meisten großen Programmpakete (z. B. dBase) geben in der Installationsanleitung an, auf welche Zahl FILES gesetzt werden soll. Es sei noch kurz erwähnt, daß Files nur für die neueren – heute fast ausschließlich genutzten – Unix-ähnlichen DOS-Funktionen zuständig ist. Für ältere Programme muß die Einstellung mit FCBS erfolgen. Weitere Erklärungen würden an dieser Stelle zu weit führen.

BUFFERS legt die Anzahl interner Puffer (a 512 Byte) fest, die bewirken, daß weniger Disketten- (bzw. Platten-)Zugriffe nötig sind, weil jeweils eine größere Datenmenge als ein Sektor in den Speicher gelesen wird. Der Sektor ist die kleinste Einheit, die mit einem Mal gelesen oder geschrieben werden kann. Möglich sind Werte zwischen 1 und 89, Standard ist 2, und empfohlen werden Werte zwischen 10 und 20, da bei größeren Zahlen nicht nur zu viel Speicherplatz verloren geht, sondern die Zugriffe auch wieder langsamer werden (aufgrund der langen Suche in den Puffern).

Zum Einstellen landesüblicher Schreibweisen dient der **COUNTRY**-Parameter. So schreiben die Amerikaner das Datum mit Schrägstrichen (06/18/90), die Europäer mit Punkt (18.06.90). Unterschiedlich sind auch die Zeitangaben (2:00 pm bzw. 14:00) und die Abtrennung der Tausender bei Zahlenkolonnen (mit Punkt bzw. Komma). Standard ist die amerikanische Einstellung (Kennung 001). Mit

CONTRY = 049

wird die deutsche Schreibweise eingestellt. Die Länderkennungen entsprechen der internationalen Telefonvorwahl (049 für Bundesrepublik Deutschland). Manchmal wird auch die 037 (für DDR) angeboten. Beachten Sie bitte, daß die Anwendung von COUNTRY keinen Einfluß auf die Tastaturbelegung und die Benutzung der Umlaute hat.

Für alle Geräte (Konsole, Drucker, Laufwerke, serielle Schnittstelle) lädt DOS beim Start eigene Steuerprogramme (Treiber, englisch: device driver). Der Nutzer hat die Möglichkeit, die Installation zusätzlicher Treiber zu veranlassen. Dafür wird der **DEVICE**-Parameter verwendet. Ziel ist entweder der Anschluß zusätzlicher Geräte oder das Ersetzen vorhandener Treiber durch einen eigenen (besseren). Über diese Problematik informieren auch MP 11/89, Seite 328, und 12/89, Seite 359. Wichtige zusätzliche Treiber sind ANSI.SYS (im nächsten Teil beschrieben), MOUSE.SYS zur Bedienung der Maus und VDISK.SYS zum Anlegen einer RAM-Disk. Der DEVICE-Parameter kann mehrmals in der CONFIG.SYS erscheinen, je nachdem, wie viele zusätzliche Treiber eingebunden werden sollen. Erfordern die Treiber zusätzliche Parameter, so können sie in der Zeile mit angegeben werden, beispielsweise

device = vdisk.sys 128 512 24

für eine RAM-Disk mit 128 KByte Kapazität, 512 Byte Sektorlänge und 24 Hauptverzeichniseinträgen. Welche Parameter für jeden einzelnen Treiber möglich sind, muß den jeweiligen Handbüchern entnommen werden. Der Eintrag **LASTDRIVE =F** legt fest, daß F: das letzte mögliche logische Laufwerk sein soll. Standardeinstellung ist E:. Dieser Parameter steht im Zusammenhang mit der Möglichkeit, ein physisches Laufwerk mit verschiedenen Formaten als logische Laufwerke zu nutzen sowie mit dem SUBST-Befehl (wird in einem späteren Teil besprochen) und Laufwerksumweisungen im Netzwerk. Für jedes Laufwerk wird Speicherplatz für die Verwaltung durch DOS reserviert, vergeben Sie also nicht mehr Platz als nötig. Werden weniger Laufwerke eingetragen, als tatsächlich vorhanden, so wird die Angabe ignoriert.

Zwei Parameter sind nicht verwendet worden: Mit **BREAK** kann die Prüfung auf <CTRL> <BREAK> zum Programmabbruch ein- und ausgeschaltet werden (exakt die gleiche Wirkung, wie das gleichnamige DOS-Kommando), und mit **SHELL** wird ein eigener Kommandointerpreter festgelegt, sofern nicht COMMAND.COM verwendet werden soll. Alternativ lassen sich damit auch Änderungen in der Installation von COMMAND.COM bewirken, etwa ein Verzeichnis, aus dem er stets geladen werden soll, oder die Größe der Programmumgebung.

Nicht näher beschrieben werden in diesem Abschnitt die Konfigurationsparameter STACKS (zur Festlegung der Größe der Interruptstacks) und DRIVEPARM (zur Änderung der Laufwerksparameter). Gegebenenfalls müssen Sie im Handbuch nachschlagen. Ab Version 4.00 kommen INSTALL (zum Installieren residenter Programme), REM (für Kommentare innerhalb von CONFIG.SYS) und SWITCHES (zur Unterdrückung erweiterter Tastenfunktionen) hinzu. Es ist auch jetzt schon möglich, mit REM Kommentare einzufügen, damit wird aber eine Fehlermeldung verursacht. REM (von Remark) erinnert den Nutzer an seine Absicht, einen Kommentar eingefügt zu haben.

Werden Eintragungen in der CONFIG.SYS geändert, so muß das System auf jeden Fall neu gestartet werden (<CTRL> <ALT> <DEL> oder Reset-Taste), da die Konfigurationsdatei nur zu diesem Zeitpunkt beachtet wird. Auch ein Entfernen der so installierten Treiber ist grundsätzlich nur durch den Neustart möglich. Da dies oft falsch verstanden wird, sei noch einmal darauf hingewiesen, daß die Angaben in der CONFIG.SYS keine ausführbaren Befehle sind und deshalb nicht auf Kommandoebene gestartet werden können.

wird fortgesetzt

Schnittstellen von Turbo-C zu MS-DOS

Prof. Dr. Horst Beidatsch, Dr. Dieter Simon, Freiberg

Der folgende Artikel wendet sich an all diejenigen, die mit der Sprache Turbo-C (Version 1.5) die Vorzüge des Betriebssystems MS-DOS (Version 3.2) nutzen wollen. Insbesondere werden die dazu notwendigen Software-Interrupts und Registerstrukturen vorgestellt und an einem Beispiel erläutert.

Prinzipiell sind die vom DOS bereitgestellten Funktionen für die File-, Geräte-, Speicher- und Prozeßsteuerdienste in beliebigen Programmsprachen ausnutzbar, wenn diese über eine definierte Registerstruktur verfügen und in der Lage sind, Software-Interrupts auszuführen. Die Programmiersprache Turbo-C verfügt über solche Strukturen und gestattet somit eine maschinennahe Programmierung ohne Einschränkungen und ist in der Lage, kompakten effizienten Code zu generieren. Der Interrupts wird man sich immer dann bedienen, wenn man die Dienste und Vorzüge des Betriebssystems in Anspruch nehmen will und für diese Funktionen in der benutzten Sprache keine Möglichkeiten der Nutzung implementiert sind. Das Interesse an DOS-Funktionen entsteht bzw. steigt in der Regel mit

- der Projektgröße und damit verbundenen Problemen des Projektverwaltens
- dem Streben nach einer anwenderfreundlichen, sicheren und damit *defensiven* Programmierung
- dem Einbinden von Grafikelementen.

Das Schreiben von Dienstleistungsprogrammen verlangt ebenfalls die Nutzung von DOS-Funktionen.
Schließlich berührt eine dritte Problematik die Nutzung und Steuerung der Hardware-Komponenten und den Zugriff auf Datenträger; nicht zuletzt unter Berücksichtigung des Aspektes der Datenfernübertragung.
Turbo-C kann all diesen Anforderungen gerecht werden, ohne daß man auf das doch sehr aufwendige Programmieren in Assemblersprache zurückgreifen muß, auch wenn das Zeigerkonzept dem Beginnenden anfangs etwas schwerfällt.
Bei der Programmierung von Software-Interrupts kann es allerdings zu Portabilitätsproblemen kommen. Ein typischer Fall dafür ist das direkte Beschreiben des Bildschirmwiederholspeichers. Es ist deshalb ratsam, Interrupts nur dann einzusetzen, wenn auszuführende Funktionen durch die Sprache nicht unterstützt werden oder Interrupts effektivere Möglichkeiten bieten als die Sprache (beispielsweise um File-Allocation-Tables zu lesen, Hardware- und Diskettenchecks vorzunehmen u. a.). Die Anwendung von Interrupts ist nicht zu empfehlen, wenn die benutzte Sprache selbst die gewünschten Funktionen enthält. So läßt sich beispielsweise ein File durch INT 21H mit **ah=0x41** löschen; hier ist es aber übersichtlicher und einfacher, die Anweisung

unlink (const char *filename)

zu benutzen. Mit steigender Versionsnummer, auch maschinennaher Sprachen wie Turbo-C, ist ohnehin der Trend nach maximalem Service zu verspüren. Erinnern wir uns zum Beispiel der Turbo-C Version 1.0, so war dort das Bildschirmlöschen nur unter Verwendung des BIOS-Interrupts INT 10H mit entsprechender Registerbelegung möglich, wenn nicht eine ESC-Sequenz gesendet werden sollte. Diese setzt aber ihrerseits die Existenz von ANSI.SYS voraus. Ab Version 1.5 sind viele solcher Dialogkomponenten ergänzt worden /1/.

Registerstrukturen und Software-Interrupts

Interrupts sind unterteilt in BIOS-Interrupts und in die DOS-Funktionen, die über den Sammelinterrupt (INT 21H) ansprechbar sind.
Entscheidend für die Registerstruktur ist das Include-File *DOS.H*. Der Name verweist bereits darauf, daß hier betriebssystemnahe Funktionen vereint sind. Im Unterschied zu Turbo-Pascal sind in Turbo-C Register und Segmentregister getrennt definiert, mit Ausnahme einer *gepackten* Registerstruktur.
Im einzelnen sind die Strukturen wie folgt vereinbart:

```
struct WORDREGS {
  unsigned int ax,bx,cx,dx,
  si,di,cflag,flags;
  };

struct BYTEREGS{
  unsigned char al,ah,bl,bh,cl,ch,dl,dh,
  };
```

Beide Strukturen sind zu einer Union zusammengefaßt:

```
union REGS{
  struct WORDREGS.x;
  struct BYTEREGS.h;
  };
```

Die Segmentregister sind wie folgt deklariert:

```
struct SREGS{
  unsigned int es;
  unsigned int cs;
  unsigned int ss;
  unsigned int ds;
  };
```

Die gepackte Registerstruktur schließlich sieht folgendermaßen aus:

```
struct REGPACK{
  unsigned r_ax,r_bx,r_cx,r_dx;
  unsigned r_bp,r_si,r_di,r_ds,r_es,rflags;
  };
```

Die zuletzt geschriebene Struktur erinnert an die in Turbo-Pascal notierte Deklaration. Die mit ihr im Zusammenhang stehende Anweisung trägt überdies den gleichen Namen

intr();, und ist im *DOS.H* als

intr(int int_type,struct REGPACK *preg);

deklariert.
Die Verwendung dieser Funktion ist, zumindest für den Turbo-Pascal-Programmierer, denkbar einfach. Angenommen, man steht vor der Aufgabe, eine Liste der Hardwareausstattung lesen zu müssen, etwa um die Größe des RAMs zu ermitteln. Dann könnte man, da die Prüfbytes im BIOS auf 0:0410H stehen, so vorgehen:
Der Speicherinhalt läßt sich mit

peek(0x0,0x0410)

direkt ermitteln und lesen; danach können die Bytes wie üblich weiterverarbeitet werden.
Ist diese Adresse dem Programmierer unbekannt, so ist das Peek(.,.)-Kommando nicht anwendbar. Hier hilft uns DOS durch den BIOS-Interrupt INT 11H. Vereinbart man beispielsweise

struct REGPACK rp:

dann liefert

intr(0x11,&rp);

auf rp.r_ax die Information, welche auf 0:0410H steht.
Weitere – ebenfalls insbesondere dem Pascal-Programmierer bekannte – Möglichkeiten, DOS-Funktionen in Anspruch zu nehmen, sind die folgenden /2/:

bdos(int dosfun,unsigned dosdx, unsigned dosal);
oder
bdosptr(int dosfun,void *argument, unsigned dosal);

Beispiel
Eine Funktion, die eine Zeichenkette auf den Drucker geben soll, wäre wie folgt zu programmieren:

```
#include <dos.h>
prints(s)      /* Funktion printstring*/
char *s;
{
/* DOS-FUNKTION */
/* 0x5 ⇒ Printstring */
while(*s) bdos(0x5,*s++,0);
/*\n ⇒ CR      */
bdos(0x5, '\n',0);
}
```

Der BIOS-Interrupt 0x5H läßt sich benutzen, um den Bildschirminhalt auf den Drucker auszugeben. In *DOS.H* ist nämlich deklariert:

#define geninterrupt(i)__int__(i)

Vereinbart man nun im eigenen Programm

#define dump geninterrupt(0x5)

dann wird durch den Ruf von Dump ein Bildschirmausdruck erzeugt. Diese Funktion ist unabhängig von der Bildschirmseite. Ein genereller Software-Interrupt ist unter Turbo-C durch die folgenden Funktionen ausführbar:

```
int86(int intr_nr,union REGS *inregs,
                  union REGS *outregs);
int86x(int intr_nr,union REGS *inregs,
                   union REGS *outregs,
                   struct SREGS *sregs);
```

Hierbei stellt intr_nr die jeweilige Interrupt-Nummer dar. Genereller Software-Interrupt bedeutet, daß jeder beliebige verfügbare Interrupt ausführbar ist. Im Gegensatz dazu sind die Funktionen

```
intdos(union REGS *inregs,
       union REGS *outregs);
intdosx(union REGS *inregs,
        union REGS *outregs,
        struct SREGS *sregs);
```

immer bezogen auf den Sammelinterrupt (INT 21H) ausgelegt. Damit der Gebrauch der Funktionen plastischer wird, eine Demonstration:
In der Turbo-C-Version 1.0 soll der Cursor auf eine bestimmte Zeile und Spalte positioniert werden, ohne eine vordefinierte Funktion *gotoxy*(spalte,zeile) wie sie in Version 1.5 mit *Conio.h* existiert. In der Assemblersprache wäre dann zu notieren:

```
...
MOV AH,2
MOV DH,<zeile>
```

```
MOV DL,<spalte>
MOV BH,<Bildschirmseite>
INT 10H
...
```

wobei INT 10H der Video-Interrupt ist. Die Verwandtschaft der Sprache Turbo-C zu Assembler wird an der Umsetzung des Problems deutlich:

```
...
#include <dos.h>
...
typedef union REGS REGISTERS;
main()
{
struct SREGS sregs;
REGISTERS  regs;
...
}
gotoxy(spalte,zeile)
{
REGISTERS regs;
regs.h.ah=0x02;
regs.h.dh=<zeile>;
regs.h.dl=<spalte>;
regs.h.bh=<bildschirmseite>;
int86(0x10,+regs,+regs);
}
```

Es ist zweckmäßig, die Union REGS einer Typdefinition typedef zu unterwerfen. Ist auf einem anderen Rechner diese Union anders bezeichnet, ändert man lediglich diese Deklaration und spart mitunter erheblichen Anpassungsaufwand.
Die Funktion, die sich auf den INT 21H bezieht, verhält sich natürlich wie der Interrupt selbst, da sie eine Untermenge darstellt. Nehmen wir an, wir wollten die Funktion chdir
chdir (const char*path)
aus *DIR.H* nachempfinden, so müßte man programmieren:

```
...
regs.h.ah=0x3B;
regs.x.dx=FP_OFF(pfad);
sregs.ds=FP_SEG(pfad);
intdosx(+regs,+regs,+regs);
...
```

Natürlich ist es möglich, die letzte Zeile durch die folgende zu ersetzen:

```
int86x(0×21,+regs,+regs,+sregs);
```

Turbo-C ist somit tatsächlich maschinennah. Ab der Turbo-C-Version 2.0 ist zudem noch das direkte Einbinden von einzelnen Maschinenbefehlen durch das Kommando **asm** oder die Funktion __emit__ möglich. Man kann also die Funktion gotoxy() durch das Assemblerbeispiel ersetzen, wobei jedem Befehl **asm** vorangesetzt werden müßte. Allerdings vergrößert die direkte Verwendung von Assemblercode auch die Fehlerquellen. Es ist unserer Meinung nach besser, den Assembler-Modul zu separieren und alle nötigen Vereinbarungen durch TCC.EXE mit der Option -S (Assembler-Code erzeugen) erstellen zu lassen.
Im ersten Beispiel wurde das Kommando peek (.,.) folgendermaßen demonstriert:

```
peek(unsigned segment,unsigned offset);
poke(unsignet segment,unsigned offset,
                              int value);
```

Im Falle Byte sind **peekb(.,.)** und **pokeb(.,.,.)** zu verwenden. Von Basic her bekannt und unter Turbo-Pascal durch das Array mem [.:.] repräsentiert, spielen diese Kommandos durchaus eine Rolle. Wir möchten Ihnen die Verwendung der Kommandos am Beispiel der Erzeugung einer Laufschrift demonstrieren. Hierbei gilt es zwei Probleme zu lösen:
① Ermitteln der Adressen des Bildschirmwiederholspeichers, die von der Grafikkarte abhängen;
② Erzeugung der Laufschrift selbst.
Das beste Verfahren, die Grafikkarte sicher zu ermitteln, ist ein Hardware-Test über die *Equipment*-Liste. Pascal-Programmierer lösen das Problem durch das Lesen der Adresse des Eintrages der Grafikkarte im Adreßraum durch **mem [$0000:$0449]**. C-Programmierer könnten das gleiche durch peek(0x0,0x0449) erreichen.
Die Prüfbytes des Rechners sind direkt auf 0:0410 ansprechbar und haben folgenden Aufbau:

Position	15 14	13	12	11 10	9	8	7 6	5 4	3 2	1	0
	prt		j	aux			drv	vid	RAM		

prt : Printer installiert
j : Game Adapter
aux : RS232 Ports
drv : Diskettenlaufwerke-1
vid : Video Mode
RAM: Planar-RAM in 16 KBytes (Speicherbänke)

Hier ist also das niederwertige Halbwort mit den Bits 5..4 interessant. Die Prüfbytes werden mit dem schon bekannten Interrupt INT 11H:intr(0x11,&rp) ermittelt und enthalten inrp.r_ax das Ergebnis. Gefragt ist natürlich die Bitbelegung. Unter Turbo-C ist das kein Problem. Das niederwertige Halbwort verbindet man einfach mit dem Operator & über die Zweierpotenz.

Beispiel:

```
printf ((rp.r_ax & 1);
if ((rp.r_ax & 2)>0) printf("1");
  else printf("0");
  ...
  if ((rp.r_ax & 128)>0) printf("1");
    else printf("0");
```

Um das höherwertige Halbwort abzuarbeiten, ist der Wert von rp.r_ax ganzzahlig durch 256 zu dividieren:

```
rp.r_ax=(int) rp.r_ax/256;
```

Dann wird wie beim niederwertigen Halbwort vorgegangen. Den eigentlichen Bildschirmmodus ermittelt man wie folgt:
Ist *Hwconf* der Wert von rp.r_ax vor der ganzzahligen Division, und steht Vid im niederwertigen Halbwort, dann ist klar, daß der eigentlich zu überprüfende Interrupt sich aus der konjunktiven Addition von Bit 4 und Bit 5 ($2^4=16$, $2^5=32$ ergibt:

```
test=(hwconf&16)+(hwconf&32);
```

Da *Vid* 2 Bit belegt, gibt es 2^2 Fälle des Zustandes beider. Die Prüfziffer ergibt sich nun aus der Zweierpotenz des niederwertigen Bits, also 2^4. Bild 1 zeigt eine mögliche Programmvariante.
Beim zweiten Problem ist zu beachten, daß die Laufschrift auf einer bestimmten Zeile in einer definierten Spalte zu positionieren ist, daß die Laufschrift (die Message) eine bestimmte Länge hat und eine Geschwindigkeit aufweisen muß. Aus der Analyse der Grafikkarte ergibt sich die Adresse, auf die die Laufschrift zu speichern ist. Ist eine Farbkarte vorhanden, so lautet die Adresse des Bildschirmwiederholspeichers B800H, anderenfalls B000H.
Alles andere kann der Funktion Laufschrift (Bild 2) entnommen werden (vgl. auch /3/).
In unserem Beispiel wurde eine Color-Karte unterstellt.
Es zeigt recht anschaulich das Zusammenwirken von Interrupts und den direkten Zugriff auf den Adreßbereich des Hauptspeichers.

Weitere Anwendungsfälle

Es gibt natürlich auch ernsthaftere Fälle, Interrupts zu gebrauchen. Beispielsweise das Problem, ein Cross-Reference-Programm zu entwickeln, welches alle Include-Files berücksichtigt. Ist ein File Read-only, kann man nicht ohne weiteres eine solche Cross-Reference erstellen. Es muß also gewährleistet werden, daß die Files bei der Bearbeitung den Read/Write-Status besitzen und nach Bearbeitung wieder den ursprünglichen Status annehmen. Tafel 1 zeigt die dabei möglichen Dateiattribute.
Alle Attribute sind additiv anwendbar.
Mit dem Interrupt 0×43H auf dem Register

```
switch(test) {
       case 0:printf("ungueltig");break;
       case 16:printf("Color-Card 40x25 monochrom");break;
       case 32:printf("Color-Card 80x25 monochrom");break;
       case 48:printf("Monochrom-Card 80x25");break;
       default:break;
              }
```

Bild 1 Menü zur Adreßbestimmung des Bildschirmwiederholspeichers

```
void laufschrift (x,y,laenge,geschwind)
int x,y,laenge,geschwind;
{ unsigned int merker,a;
            int i;
  do {
       merker=peek(0xb800,(160*(y-1)+2*(x-1)));
       for (i=x;i<=x+laenge-1;i++){
           a=peek(0xb800,(160*(y-1)+2*(i)));
           poke(0xb800,(160*(y-1)+2*(i-1)),a);
           poke(0xb800,(160*(y-1)+2*(x+laenge-1)),merker);
           delay(geschwind);
                                       }
     } while(!kbhit());
}
```

Bild 2 Programmausschnitt Laufschrift

ah über INT 21H aber ist die Klärung des Problems einfach. Ähnlich unkomplizierte Herangehensweisen ergaben sich bei solchen Problemen wie dem Bestimmen von Filegrößen und dem Lesen der FAT, um Aussagen wie bei CHKDSK.EXE zu erreichen. Ebenso ist es relativ einfach möglich, die Diskettenkapazität zu prüfen, formatierte Type-Funktionen zu erzielen, bestimmte Unix-Funktionen (z. B. GREP a file) zu implementieren. Directories anzulegen und zu löschen, Formatierungen für verschiedene Disketten-Formate vorzunehmen u. a. Ein weiteres Anwendungsbeispiel könnte auch das Katalogisieren von Disketten sein.

Tafel 1

	Dual 7 6 5 4 3 2 1 0	Hexa-dezimal	Bedeutung
Bit-muster	0 0 0 0 0 0 0 0	00	R/W
	0 0 0 0 0 0 0 1	01	R/O
	0 0 0 0 0 0 1 0	02	HIDDEN
	0 0 0 0 0 1 0 0	04	SYS
	0 0 0 0 1 0 0 0	08	Volume
	0 0 0 1 0 0 0 0	10	(Sub-) Directory
	0 0 1 0 0 0 0 0	20	Archiv
	0 0 1 1 1 1 1 1	3F	any file

Literatur

/1/ Schildt, H.: Advanced TURBO-C. Borland-Osborne/ McGraw-Hill Book Company, California 1987

/2/ Hartwig, O.: TURBO-PASCAL für Insider. Markt und Technik Verlag AG, Haar bei München 1987

/3/ Kalfa, W.: Das Betriebssystem MS-DOS. Mikroprozessortechnik Berlin 2 (1988) 12, S. 356

/4/ Kernighan, B. W. Ritchie, D. M.: Programmieren in C. Carl-Hanser-Verlag München, Wien 1983

/5/ Hübener, J.: MS-DOS. Verlag Technik, Berlin 1988

✉ KONTAKT ☎

Bergakademie Freiberg. Zeunerstr. 8–10, Freiberg, 9200; Tel. 51 26 74

Unix-Grundlagen

Alles zur gleichen Zeit

Wolfgang Börner

Die steigende Popularität von Unix ist allgemein anerkannt. Dieses Multiuser- und Multitasking-Betriebssystem hat langjährige und treue Anhänger unter professionellen Anwendern und vor allem an den Universitäten. (jw)

Häufig hört man die lapidare Feststellung, die Ziele anderer Betriebssystemhersteller seien von Unix bereits seit Jahren erreicht; an die bislang noch unhandliche Benutzeroberfläche haben sich die Vertreter dieser Meinung ohnehin gewöhnt.

Wo liegen die Stärken und Schwächen dieses in den frühen siebziger Jahren in den Bell Laboratorien in den USA entwickelten und seither fortlaufend verbesserten Betriebssystems? Welche Chancen für eine weitere Verbreitung gibt es, und wer setzt es bevorzugt ein? Diese und eine Reihe anderer Fragen haben CP veranlaßt, die Vergangenheit und Zukunft von Unix zu durchleuchten und offenzulegen.

Besonders interessant scheint es auch, die Entwicklungen um OS/2 zu betrachten, da mit diesem jüngsten Kind der Microsoft-/IBM-Softwareschmiede ein weiterer Bewerber um die Käufergunst im Disput „Multiuser kontra Netzwerke" auf der Bühne erschienen ist. Die kommenden Monate werden grundlegende Entscheidungen bringen, wenn der Burgfrieden in den Reihen der Unix-Organisationen und Normierungsausschüssen um eine einheitliche grafische Benutzeroberfläche für Unix anhält. Für diese Entwicklung müssen die passionierten Mini-Spezialisten dem Mitbewerber OS/2 dankbar sein. Auch die wachsende Zahl der 386er-PCs im Markt und daran angepaßte Unix-Versionen wie SCO Unix V/386 tragen zur Steigerung der Nachfrage bei.

Mehr als 3000 kommerzielle Anwendungen unter Unix sind zur Zeit in den Katalogen der Systemhersteller aufgelistet, nahezu für jeden Bedarf gibt es eine große Auswahl. Kann Unix die Welt der Desktop-PCs erobern?

Unix ist an Multifunktionalität kaum zu überbieten. Kein anderes Betriebssystem läuft auf so vielen Prozessoren und Computersystemen. Rund 20 Jahre hat diese Entwicklung von Ken Thompson und Dennis Ritchie hinter sich. Nun ist mit OS/2 ein Mitbewerber erwachsen, der mit seiner „eingeschränkten" Verwendbarkeit ausschließlich für Intel-Prozessoren gerade deswegen zu einem ernsthaften Konkurrenten wird. Auch die bereits etablierten Netzwerkhersteller machen Front gegen dieses ursprünglich „elitäre Hochschul-Betriebssystem". Welche Chancen hat Unix auf dem PC? Liest man die Einführungen zu Unix-Büchern, so erscheint das Betriebssystem als eines der „einflußreichsten und populärsten überhaupt". Schätzungen zufolge sollen derzeit rund eine Viertel Million Computer unter Unix arbeiten. Zum Vergleich: Man schätzt die Zahl der MS-DOS-Installationen auf ca. 30 Millionen. Aber MS-DOS-Anwender werden manchmal von den wahren Unix-Enthusiasten noch immer nicht für voll genommen. Als ebenbürtiger Gegner aber wächst OS/2 heran, das Kenner für den „Renner" der neunziger Jahre halten.

Der alte Streit zwischen den Bedienern von Mainframe- und Minicomputern einerseits und den PC-Anwendern andererseits verläuft, so scheint es, mehr und mehr zugunsten der Personal-Computer. Die bisher gepriesenen Vorzüge der Groß-EDV: virtuelle Adressierbarkeit, große Festplatten und konsequente Kontrollmöglichkeiten der Geschehnisse in der Rechnerumgebung, ringen dem PC-Benutzer nur noch ein müdes Lächeln ab. Moderne Prozessoren, adressierbarer Hauptspeicher im Gigabereich und räumlich kleine Massendatenspeicher mit großen Kapazitäten lassen die Systemadministratoren der Mittleren Datentechnik (MDT) aufhorchen. Das alles gibt es in kleinen Gehäusen, die auf oder neben dem Schreibtisch Platz haben.

Es bleiben die Zauberworte Multitasking- und Multiuser-Betrieb. Beide Voraussetzungen bringt Unix von Haus aus mit. Nachdem die Betriebssystemhersteller erkannt haben, daß die bisher belächelten PCs kaum mehr von Workstations zu unterscheiden sind, beginnen sie sich dem Fortschritt anzupassen; allen voran die Santa Cruz Operation mit ihrem SCO Unix V/386.

Die Gegner von Unix setzten auf Vernetzung, auf lokale Netzwerke (LAN) für Einzelplatz-PCs. Auf das eigentliche Betriebssystem wurde eine Netzwerkshell gesetzt, die zu einem eigenen Netzwerkbetriebssystem anwuchs. Mit dem Auftreten von OS/2 wird diese Konfrontation allerdings um einen Gegenspieler auf der Seite der Multiuser- und Multitasking-Betriebssysteme verstärkt. Die eigentliche Fragestellung des Anwenders lautet also, ob ein Netzwerk aus Einzelplätzen oder eine Multiuser/Multitasking-Umgebung für seinen Einsatzbereich sinnvoller ist. Wo liegen für den Anwender die wesentlichen Unterschiede im Betrieb einer Multiuser- im Gegensatz zu einer Netzwerkumgebung? Grundsätzlich keine: Beiden gemeinsam ist der Wunsch, im Team zusammenzuarbeiten, um Redundanzen zu vermeiden und kostenaufwendige Geräte gemeinsam zu nutzen. Der so häufig geübte Preisvergleich ist bei der Beurteilung nur *ein* Kriterium, wichtiger erscheinen die Funktionsvergleiche.

Teamarbeit steht im Vordergrund

Für ein reibungsloses Zusammenspiel vieler Arbeitsplätze sind eine Reihe von administrativen Funktionen notwendig, unter anderem die Überwachung des ordnungsgemäßen Verhaltens der beteiligten Peripheriegeräte und die Vergabe und Kontrolle der Zugriffsrechte.

☐ Sind die Daten vor unrechtmäßigem Zugriff geschützt?

☐ Wie reagiert der Computerverbund auf Ausfall eines Teilnehmers?

☐ Wie viele Versionen einer Anwendungssoftware müssen gekauft werden?

☐ Wer verwaltet den zur Verfügung stehenden Arbeitsspeicher mit welchen Prioritäten?

☐ Wann entstehen Last- und Performance-Probleme?

Diese und viele andere Fragen müssen beantwortet werden, um nicht nach dem Kauf einer Anlage vor zum Teil kostenträchtigen Problemen zu stehen.

Wichtig ist, wann die Entscheidung zwischen Netzwerk oder Multiuser-Umgebung gefällt

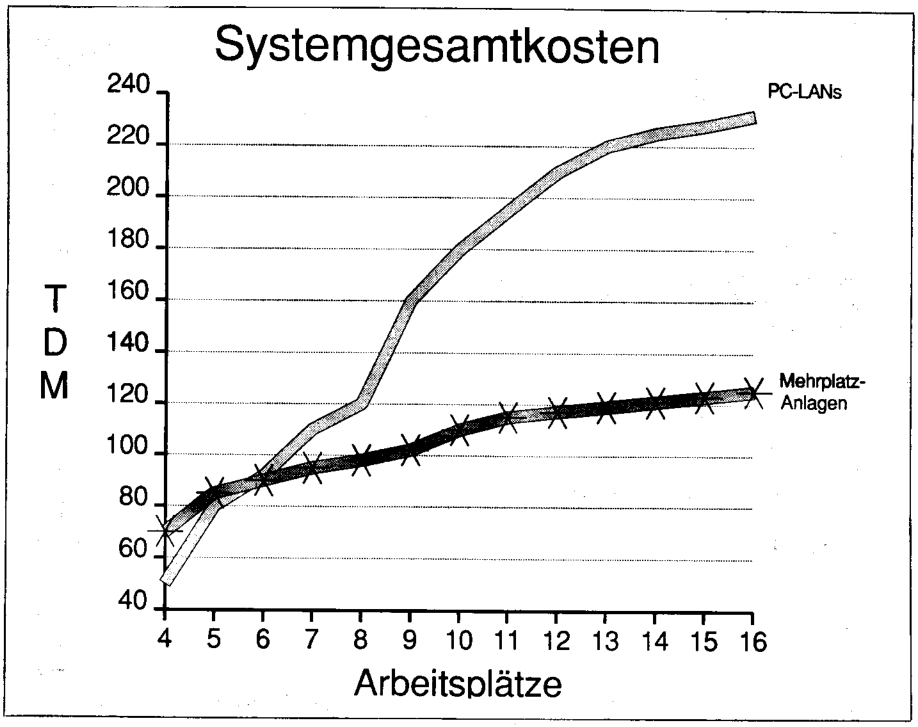
wird. Wenn alle Arbeitsplätze bereits mit PCs ausgerüstet sind, fällt die Entscheidung für ein Netzwerk leicht. Arbeiten hingegen bisher nur wenige Mitarbeiter an einem Personal-Computer, so ist die Wahl offen.

Ein weiterer Punkt sind die einzusetzenden Anwendungen. Sollen generell mehrere Programme individuell benutzt werden, oder beschränkt sich die Auswahl der Software auf einige von allen genutzten Programmpakete? Wenn viele Anwendungsprogramme für völlig unterschiedliche Arbeiten benötigt werden, so empfiehlt sich ein Netzwerk, um die einzelnen Anwender ungestört voneinander arbeiten zu lassen. Es gibt auch eine Reihe von Anwendungen, die kontinuierlich abgearbeitet werden müssen. In diesen Fällen wären möglicherweise alle anderen Teilnehmer in einer Multiuser-Umgebung „ausgesperrt". Als Gegenstück dazu sind die typischen Batch-Verarbeitungsläufe bei Finanzbuchhaltungsprogrammen o. ä. auf einer Multiuser-Anlage gut aufgehoben. Generell gilt, daß heterogene Anwendungen in einem Netzwerk besser aufgehoben sind. Besonders auffällig wird dieser Grundsatz im Workgroup Computing, wenn z. B. Publikationen mit Electronic Publishing erstellt werden. Dabei bevorzugt der DTPler einen Macintosh, die Tabellen werden auf einem PC bearbeitet, und die endgültige Montage erfolgt auf einem Unix- oder OS/2-PC. Der Datenaustausch erfolgt über ein Netzwerk, so daß jeder Anwender in seiner bevorzugten Umgebung die erforderlichen Leistungen bereitgestellt bekommt. Die Entscheidung für einen Unix-basierten (386er-)PC liegt in der für dieses Betriebssystem angebotenen hervorragenden Electronic-Publishing-Software.

Neben der Abarbeitung im Stapel wird häufig eine Hintergrundverarbeitung gewünscht. Einzelplatzsysteme bieten dazu bisher keine vernünftigen Möglichkeiten, wenn beispielsweise große Datenbestände abgefragt oder ausgewertet werden sollen. Bei Vordergrundprozessen kann unter Umständen während der Abarbeitung der Arbeitsplatz blockiert sein. In allen modernen Unix-Versionen sind sowohl komfortable Hintergrundverarbeitungs- als auch Spooling-Routinen vorhanden.

Die quasi gleichzeitige Nutzung von Daten wird als Data Sharing bezeichnet. Dabei unterscheidet man zwei grundlegende Möglichkeiten: den Zugriff auf Dateien (Files) oder einzelne Datensätze (Records). Zur Vermeidung von unkontrollierten Kollisionen müssen entsprechende Schutzmechanismen vorgesehen sein. Unix wurde von Anfang an mit vielfältigen und wirksamen Schutzvorrichtungen ausgestattet, die modernen Netzwerk-Betriebssysteme bieten inzwischen jedoch gleiche Schutzmechanismen. Generell wird Unix einzelne Records aus einem File rascher an den angeschlossenen Terminals bereitstellen können. Anders hingegen sieht die Situation aus, wenn mehrere Anwender in einer Tabellenkalkulation in derselben Tabelle arbeiten. Die Rekalkulation einer großen Tabelle erfordert viel CPU-Zeit und bringt in einem Multiusersystem lange Reaktionszeiten für die beteiligten Anwender. Bei überwiegendem Einsatz von Kalkulationssoftware sind vernetzte PCs vorteilhafter.

Ein besonders wichtiges Kriterium bei der Entscheidung zwischen Unix und seinen Alternativen ist die Frage der Standardisierung. Hier bietet Unix erhebliche Vorteile, wenn man die tatsächliche Standardisierung des



Betriebssystemkernes betrachtet. Nicht von ungefähr weist es die meisten Installationen auf völlig unterschiedlichen Computersystemen auf. Nur ein kleiner Systemkern (Nucleus) ist maschinenabhängig, alle anderen Programmteile sind in systemunabhängigen Programmiersprachen geschrieben, in der Regel in C. Zusätzlich ist eine beschriebene Zahl von Kommandos allen Unix-Versionen gemeinsam. Der Käufer einer Unix-Anlage hat also die Gewißheit, daß seine Programme relativ leicht portiert werden können. Daß im täglichen Soft- und Hardwareleben trotzdem ein erheblicher Aufwand getrieben werden muß, liegt in der Regel an den notwendigen Anpassungen an die neue Peripherie.

Die Übertragbarkeit von Software zwischen unterschiedlichen Netzwerken ist weitaus problematischer, es gibt hier nur die sogenannten „De-facto-Standards" der jeweils durch die Zahl ihrer Installationen dominierenden Hersteller. Auch die Kompatibilität unter OS/2 ist sehr fragwürdig, wenn es beispielsweise um den Datenaustausch zwischen Datenbankprodukten verschiedener Hersteller geht. SQL ist lediglich ein Quasi-Standard. Obwohl Microsoft die Schnittstellen zum OS/2 verbindlich für alle Anwendungsprogramme beschreibt, gibt es an verbindlichen Regeln für die Kommunikation der Applikationen untereinander noch einiges zu entwickeln.

Mit dem Aufkommen der 386er-Prozessoren hat sich die Marktsituation für Unix auf dem PC wesentlich verändert. Bislang waren LANs die preiswertere Alternative zu Multiuser auf Superminis. Heute haben zum einen

Merkmal System	PC-LAN	PC-LAN	Mehrplatzsysteme
Anzahl der Arbeitsplätze	bis 10	> 10	5 bis 100
Einsatzbereich	Kommunikation und Ressoursharing	Kommunikation und Ressoursharing	Multiuser-Anwendungen
Prozessor	80286, 80386	80286, 80386	80286 (Xenix), 80386 (Unix)
Plattenkapazität dezentral Server	 bis 40 MByte bis 160 MByte	 bis 40 MByte bis 900 MByte	 – bis 1400 MByte
Plattenzugriffe zentral	gelegentlich	häufig	oft
Betriebssystem	z. Zt. DOS, Singletasking oder OS/2, Multitasking	z. Zt. DOS, Singletasking oder OS/2, Multitasking	Unix (Xenix), Multitasking
Anwendungen	Standard, Singleuser	Standard, Singleuser	Standard, Spezial, Multiuser/Timesharing
Antwortzeiten System: abhängig von	Anwendungen, Prozessor, Hauptspeicher	Anwendungen, Prozessor, Hauptspeicher	Anwendungen, Prozessor, Hauptspeicher, Anzahl der User
zentrale Platte: abhängig von	Server/Prozessor, Netz, Anzahl der Zugriffe	Server/Prozessor, Netz, Anzahl der Zugriffe	I/O-Prozessor, Zahl der Zugriffe
Datensicherheit -schutz	gut gering	gut gut	gut gut
Übertragung Kabel Geschwindigkeit Protokoll Entfernung	 Basisband ≦ 3 MBit/s Standard ≦ 300 m	 Basisband ≦ 10 MBit/s Standard ≦ 300 m	 diverse ≦ 100 kBit/s eigene ≦ 100 m, 2500 m mit Verstärker
Kosten pro Arbeitsplatz	15 000 D-Mark	20 000 D-Mark	10 bis 40 000 D-Mark, abhängig von der Anzahl der AP

Die Gegenüberstellung zeigt deutlich die Gemeinsamkeiten und Unterschiede von LANs und Mehrplatzsystemen

die steigenden Installationen den Verkaufspreis für PC-Unix sinken lassen bzw. die Portierung forciert, zum anderen ist die Performance der PCs derartig angehoben worden, daß ein direkter Vergleich zu Minirechenanlagen möglich wurde. Durch den vergrößerten und virtuell verwalteten Hauptspeicher sind PCs zu einer Plattform für die Applikationen der Minis herangewachsen. Im Zuge der weiteren Entwicklung um den 486er mit 15 MIPS (Million Instructions per Second), 256 MByte Speicher und bis zu 4 GByte Massenspeicherkapazität über einen 32 Bit breiten Bus, haben praktisch alle Anlagen der MDT und auch einige preiswerte Mainframes das Nachsehen in einem Vergleich. Das bedeutet, daß die Preise für Multiuser-Anlagen merklich gefallen sind; in der Regel liegen sie sogar unter dem für ein Lokales Netzwerk mit gleich vielen Anwendern. Wo nun liegt der Weisheit letzter Schluß? Wie so oft in der Mitte. Beide Systemwelten haben ihre bevorzugten Einsatzbereiche. LANs sind vorzugsweise einzusetzen bei heterogenen Anwendungsprofilen, bei Zugriffen auf viele verschiedene Dateien und bei vorwiegend und direkt abgearbeiteten Prozessen. Multiuser-PCs eignen sich besonders bei homogenen Anwendungsprofilen, zum Beispiel beim Zugriff auf Datensätze in wenigen (großen) Dateien, wenn die Verlagerung von Prozeßteilen in den Hintergrund gewünscht wird und weil die Gesamtkosten niedriger sind. Jede Systementscheidung erfordert also eine gründliche Analyse der auszuführenden Arbeiten, wobei möglicherweise ein Verbund aus LAN und Mehrplatzanlage den größtmöglichen Erfolg verspricht. Nicht zuletzt das steigende Softwareangebot für Unix im kaufmännischen/verwaltenden Bereich spricht für eine derartige Lösung. Vor allem aber sind alle Türen für künftige Entwicklungen offen gehalten.

Bei der Entscheidungsfindung helfen möglicherweise auch die Entwicklungen im Softwaremarkt. Unix krankt am mangelhaften Komfort in der Bedienung und an den Querelen zwischen den Standardisierungsgruppierungen. Der verfügbare Pool an Anwendungsprogrammen ist allerdings um ein Vielfaches größer als derzeit für OS/2 angeboten werden. Andererseits ist für die kommenden zwei bis drei Jahre in diesem Bereich eine stürmische Entwicklung zu erwarten, die wohl auch vor den Verkaufszahlen für LANs nicht Halt machen wird. Das Betriebssystem der Zukunft für PCs heißt sicher OS/2. Trotz aller Schwierigkeiten, angekündigte Fertigstellungstermine zu halten, erwarten Branchenkenner den Durchbruch für 1991. Bis dahin wird es auch ausreichend unterstützte Anwendungsprogramme geben.

OS/2 ist im endgültigen Entwicklungsstadium, ergänzt um den LAN-Manager, auch ein vernünftiger Ersatz für vernetzte Einzelplatz-PCs. Durch seine enge Bindung an die Prozessoren von Intel wird es für Softwarehäuser weniger aufwendig sein, Programme für den Masseneinsatz zu entwickeln, als für verschiedene Rechnersysteme unter Unix, wenn sie dabei auf vergleichbare Stückzahlen kommen wollen. Für die kommenden Jahre ist Unix jedoch aufgrund seines umfassenden und individuellen Softwareangebotes ein ernstzunehmender Partner an der heißumkämpften Front der PC-Betriebssysteme, zumal in den kommenden Monaten die Entscheidung für eine einheitliche grafisch orientierte Benutzeroberfläche in Aussicht steht. Neben den amerikanischen, englischen und schwedischen Behörden haben sich inzwischen auch einige Bundesländer in Deutschland für Unix als Voraussetzung für einen Hardwarekauf entschieden.

Quellenhinweise:
Dokumentationen der Firma SCO, Deutschland; Dokumentation Online '89: PC-LAN vs. Mehrplatzsysteme, J. Peulen München

---

Nachdruck aus Computer Persönlich, Heft 4/1990

Computermesse für West-Berlin und DDR

Am 7. Juli 1990 veranstaltet die Fischel GmbH in West-Berlin bei Schultheiss in der Hasenheide 23–31 (U-Bhf. Hermannplatz) in der Zeit von 8 bis 18 Uhr die „1. Private & gewerbliche Computermesse für Berlin und Brandenburg".

Auf der Verkaufsmesse werden neue und gebrauchte Computer, Software und Literatur gehandelt; das heißt, Private sowie Berliner und auswärtige Händler können ihre Produkte zum Kauf anbieten.

Der Eintrittspreis beträgt 7,– DM.

Bei Rückfragen wenden Sie sich bitte an Herrn Hüttinger, Tel. Leipzig 32 96 92.

Wegbereiter der Informatik

FRIEDRICH ADOLF WILLERS

∗1883 Bremervörde
† 1959 Dresden

F. A. Willers hat 1903–06 in Jena und Göttingen Mathematik und Physik studiert, u. a. bei C. Runge (1856–1927). Ab 1905 war er am Mathematischen Institut der Universität Göttingen bei C. Runge Hilfsassistent und wurde dort 1906 promoviert. In den Jahren 1908–11 war er Assistent für Darstellende Geometrie und Graphische Statistik an der Technischen Hochschule Danzig. Danach unterrichtete er über zehn Jahre an höheren Schulen, zuletzt am Mommsen-Gymnasium in Berlin-Charlottenburg. Im Jahre 1923 habilitierte sich Willers als Privatdozent an der Technischen Hochschule Berlin und übernahm dann eine Dozentur für Mathematik und Mechanik des Hüttenwesens. Einem Ruf an die Bergakademie Freiberg folgend, nahm er dort von 1928 bis 1934 eine ordentliche Professur für Mathematik und Darstellende Geometrie wahr und lebte danach als freischaffender Wissenschaftler. Willers' letzte Station in seinem wissenschaftlichen Leben war Dresden; an der dortigen TH hielt er ab 1939 Vorlesungen in Analysis und Angewandter Mathematik und trat als Nachfolger von Max Lagally in diesem Fachbereich 1944 eine ordentliche Professur an. Ein Jahr nach Kriegsende wurde er Direktor des Instituts für Angewandte Mathematik und war 1949–54 Dekan der Mathematisch-Naturwissenschaftlichen Fakultät.

Willers gebührt das Verdienst, wesentlich mit dafür gesorgt zu haben, daß die maschinelle Rechentechnik Einzug in die Lehre an den Hochschulen fand. Er hatte erkannt, welche Bedeutung die Rechenmaschinen für die Anwendung der Mathematik einmal haben würden und hat daraus resultierende Überlegungen und Methoden seinen Schülern schon zu einer Zeit vermittelt, als Universalrechner noch nicht verfügbar waren. Diesem Anliegen dient z. B. auch sein in Fachkreisen allgemein bekanntes Lehrbuch über „Mathematische Instrumente", das 1926 erstmals im Oldenbourg-Verlag erschien und bis 1951 mehrfach in erweiterter Auflage herausgebracht wurde. Es geht zum Beispiel weit über den Rahmen von L. Couffignals „Denkmaschinen" (Paris 1933, Stuttgart 1955) hinaus und gehört zu den frühen grundlegenden, zusammenfassenden Standardwerken der maschinellen Rechentechnik überhaupt. Darin wird unter anderem das Leibnizsche Staffelwalzenprinzip konstruktiv begründet, und es werden auch Apparate zur Flächen- und Momentenberechnung, harmonische Analysatoren, Stieltjes-Planimeter, Differentiatoren und Integraphen sowie verschiedene Formen von Datenspeichern ausführlich beschrieben. Willers hat noch einige andere Bücher zu Gebieten der praktischen Mathematik verfaßt, mit C. Runge zusammen auch ein Lehrbuch über numerische und graphische Integration gewöhnlicher und partieller Differentialgleichungen.

Willers war Mitglied der Akademie der Naturforscher (Leopoldina) zu Halle, der Sächsischen Akademie der Wissenschaften zu Leipzig und korrespondierendes Mitglied der Berliner Akademie. Im Jahre 1953 wurde er mit dem Nationalpreis der DDR ausgezeichnet und mit einem Ehrendoktorat der Technischen Hochschule Darmstadt gewürdigt.

Dr. Klaus Biener

Inline eingeschränkt

Turbo-Pascal zwingt zu Umwegen

Bernhard Brandt

Die Schnittstelle zur Maschinensprache ist in Turbo-Pascal auf dreierlei Art implementiert. Einmal in der direkten Einbindung von Routinen, die ein symbolischer Assembler erstellt hat, und via „inline", wobei Turbo-Pascal zwischen Prozeduren/Funktionen und Makros unterscheidet. Eigentlich sollten sich Inline-Makros zur Erweiterung der hardwarenahen Befehle wie „shl", „shr", „dec" und „inc" eignen. Dabei treten jedoch Schwierigkeiten auf.

Wie im Kasten zu sehen ist, läßt sich die Anweisung „rol (Variable, 1)" über einen einzigen Assemblerbefehl komplett realisieren, wobei „ffff" beziehungsweise „ff" einen Dummy für den Offset darstellt (in DS für globale Variablen, als Relativwert zum BP-Register für lokale Variablen).

In Abhängigkeit vom Typ (Byte oder Word) und dem lokalen beziehungsweise globalen Charakter dieser Variablen (Segmentspezifizierung) muß lediglich einer dieser Befehle ausgewählt werden. Damit wäre ein neuer Pascal-Befehl mit universeller Einsatzmöglichkeit wie „shl" generiert.

```
D0 06 ffff  rol byte ptr [ffff],1        global Byte
D1 06 ffff  rol word ptr [ffff],1        global Word
D0 46 ff    rol byte ptr [bp] [ffff],1   lokal  Byte
D1 46 ff    rol word ptr [bp] [ffff],1   lokal  Word
```

```
var my_special_first_global_byte,
    my_special_second_global_byte,
    my_special_third_global_byte : byte

procedure rol_only_my_special_first_global_byte_one_time;
inline ($D0/$06/my_special_first_global_byte);

procedure rol_only_my_special_second_global_byte_one_time;
inline ($D0/$06/my_special_second_global_byte);

procedure rol_only_my_special_third_global_byte_one_time;
inline ($D0/$06/my_special_third_global_byte);
```

Unter Turbo-Pascal ist dies aber nicht möglich. Der Compiler ist nicht dazu zu bewegen, den Offset einer irgendwie angegebenen Variablen selbst einzusetzen oder ihn innerhalb einer Prozedur in einer bedingten Kompilierung in Abhängigkeit des globalen oder lokalen Charakters einer Variablen zuzulassen, obwohl dies durch den Quelltext ja angegeben wird. Statt dessen sind entsprechende Befehle nur „per Hand" für eine konkrete Variable zu erzeugen.

Die abgebildeten Prozeduren enthalten zwar den gleichen Code, man muß sie aber getrennt programmieren. Dies gilt analog für lokale Variablen. Hier macht Turbo-Pascal seinem Namen keine Ehre. Die einzige Möglichkeit zur Realisierung eines „rol-Befehls" besteht in der speicherplatz- und rechenzeitfressenden „pushpop-Programmierung". Entsprechende Inline-Makros erzielen daher zum Teil schlechtere Ablaufzeiten als die Realisierung in Pascal, da sich effizienter Code mangels Informationstransfers nicht realisieren läßt.

Im Referenzhandbuch steht auf Seite 229 der Hinweis: „Was dagegen nicht funktioniert, ist eine symbolische Bezugnahme über die Parameternamen: ohne Startcode ist keine Adressierung über das BP-Register möglich." Dem kann man entgegnen, daß die hier gestellte Forderung an bessere Inline-Möglichkeiten nur die „Einfügung statt Pushen" des Offsets an eine frei wählbare Stelle des Inline-Codes und Zugang zu der im Quelltext angegebenen Information „global" oder „lokal" erfordert. (db)

Nachdruck aus c't – Magazin für Computertechnik, Heft 1/1990

Neue Electronic-Publishing-Werkzeuge

Am 16. Januar 1990 fand in Vorbereitung der internationalen Fachmesse der polygrafischen Industrie, DRUPA 1990, in Westberlin eine internationale Pressekonferenz der Firma Berthold statt.

Die H. Berthold AG hat ihren Hauptsitz in Westberlin und sieht als den Gegenstand des Unternehmens die Entwicklung, die Herstellung und den Vertrieb von Schriften, grafischen Symbolen und Bildelementen sowie damit verbundene Softwareprodukte für deren visuelle Darstellung, Wiedergabe und Speicherung an.

Neben dem Geschäftsbericht für das vergangene Jahr war für die internationale Presse die Vorstellung einiger neuer Produkte von besonderem Interesse. So sollte bereits ab Februar 1990 eine Palette neuer Produkte – darunter eine neue Generation von Workstations und ein hochauflösender Laser-Recorder – auf den Markt gebracht werden.

Gegenwärtig zeichnen sich Tendenzen ab, von dem bisher üblichen Verfahren abzugehen, die Unterlagen, Vorlagen und Zwischenergebnisse fast ausnahmslos körperlich transportiert in die nächste Prozeßstufe zu überführen. Mit Electronic-Publishing-Werkzeugen können professionelle grafische Betriebe jetzt selbst die eigenen Vorprodukte herstellen. Die Firma bietet Systeme, die an die der Auftraggeber anschließen und deren Vorprodukte weiterverarbeiten und Qualitätsdruckvorlagen herstellen.

Durch die Bearbeitung und den Transport in digitaler Form kann die Herstellzeit eines Druckprodukts sichtbar verkürzt werden. In Zukunft soll auch die Herstellung von kompletten Farbseiten mit Text, Bild und Grafik möglich sein, das heißt, durch den Anschluß eines Farbscanners können auf einer Farb-Workstation Bilder retuschiert, freigestellt und maskiert werden; die Ausgabe der kompletten Separation erfolgt über einen bildfähigen Recorder. Dieses von Berthold angekündigte und zum Teil auch schon angebotene System der Serie M ist zunehmend auch für kleinere Unternehmen interessant, nicht zuletzt wegen der Schnittstelle zu DTP-Systemen wie beispielsweise zum Apple Macintosh. Die Dateien (Text, Bild, Grafik) könnten somit komplett in der Redaktion erstellt und dann in einem System der H. Berthold AG weiterverarbeitet, gedruckt bzw. auf Film belichtet werden. Eine Übergabe und Übernahme von Dateien über andere Postdienste ist ebenfalls möglich und zunehmend sinnvoller. Von besonderem Interesse war eine Besichtigung der Technik in Aktion. Es wurden alle anfallenden Arbeiten in der Vorbereitung für den Satz vorgeführt; so beispielsweise das Bearbeiten und Spiegeln des Textes, Retuschieren der gescannten Bilder bis hin zum Verfremden und die abschließende Gestaltung einer kompletten Druckseite durch die Verwendung von Sun-Workstations, die insbesondere für die Bildverarbeitung mit speziellen Risc-Prozessoren ausgerüstet sind, auch in bestechender Geschwindigkeit (Leistung: 12 Millionen Instruktionen pro Sekunde). Die präsentierte Technik wird vermutlich nicht zuletzt wegen der Mitte 1989 vorgenommenen Schließung der Ost-Europa-Vertretung in Wien und deren Verlagerung nach Westberlin in den Druckereien und den vorgelagerten Bereichen in der DDR zukünftig eine größere Rolle als bisher spielen.

J. Hill

Computer-Club

Einzelpunktgrafik für den A 7100

Bernd Matzke, Delitzsch

Die Nutzung der Grafik am A 7100 erfordert normalerweise das Vorhandensein und die Aktivierung der Firmware und der grafischen Betriebssystemerweiterung. Dies ist aufwendig, in der Bedienung umständlich und erfordert einen hohen Programmieraufwand. Oft werden die gebotenen Möglichkeiten aber gar nicht benötigt.
Im folgenden soll daher ein einfaches Verfahren zur Realisierung einer Einzelpunktgrafik beschrieben werden. Dieses Verfahren erfordert weder die Betriebssystemerweiterung noch die Firmware.
Die Leistungsfähigkeit des Verfahrens ist begrenzt, reicht aber für einfache Darstellungen, wie die Anzeige von Meßwertreihen, Diagrammen (z. B. Netzwerkanalyse) und ähnliches völlig aus. Die Ausgabegeschwindigkeit erreicht ungefähr 40 Pixel je Sekunde.
Das Demonstrationsprogramm wurde in Turbo-Pascal erstellt, dürfte aber leicht in andere Sprachen umzuschreiben sein.
Das Verfahren setzt einen A 7100 mit Grafik-Karte voraus.
Auf der Grafik-Karte wird auch der alphanumerische Bildwiederholspeicher (BWS) pixelweise organisiert. Beim Schreiben eines Zeichens wird dessen vollständiges Pixelmuster in den BWS übertragen und nicht, wie normalerweise üblich, nur ein Byte mit dem ASCII-Wert. Wird nun das Muster eines Zeichens aus dem frei programmierbaren Zeichengenerator nach dem Übertragen in den BWS geändert, so hat das keine Auswirkung auf die Bildschirmanzeige! Der BWS übernimmt damit die Funktion eines Pixelspeichers, auf den aber nur schreibend und nur auf dem Umweg über Pseudo-Grafikzeichen mit einer Größe von 8 × 16 Bit zugegriffen werden kann.
Diese Eigenschaften eröffnen nun die Möglichkeit, im Alphanumerik-Modus eine Einzelpunktgrafik zu realisieren. Dazu wird im Speicher ein 640 × 400 Bit (32 KByte) großes Array angelegt, in dem die Grafik gespeichert wird. Jeweils 8 × 16 Bit in diesem Array können als Bitmuster eines Alpha-Zeichens aufgefaßt werden. Beim Setzen eines Pixels wird aus dem zugehörigen 8 × 16 Bit-Feld ein Alpha-Zeichen gebildet und in den frei programmierbaren Zeichengenerator geschrieben. Dieses Zeichen wird dann an der entsprechenden Stelle auf den Bildschirm ausgegeben.
Da die Einzelpunktgrafik über die Pseudografik erreicht wird, sind folgende Besonderheiten zu beachten:

- Alphazeichen und Grafik können gleichzeitig nebeneinander verwendet werden, allerdings nicht gleichzeitig auf einem Alphazeichen-Feld. Das Setzen eines Grafik-Punktes löscht das Alphazeichen an dieser Stelle und umgekehrt.
- Bildschirmattribute können verwendet werden, sie wirken aber zeichen- und nicht pixelorientiert. Ihr Nutzen ist daher (z. B. andere Helligkeit für die gesamte Grafik) bei Grafik-Anwendungen begrenzt.
- Für die Grafik wird nur ein frei definierbares Zeichen benötigt. Alle anderen 127 können für andere Zwecke verwendet werden.

Das vorliegende Demonstrationsprogramm beinhaltet die Prozedur PIXEL zum Setzen von Pixeln und die Prozedur *PRINT_GRAFIK* zur Ausgabe der Grafik auf einem Nadeldrucker (FX 1000 oder andere zu Epson kompatible).
Bei der Erprobung habe ich festgestellt, daß bei der Ausgabe von Zeichen mit ASCII-Werten oberhalb 253 durch die Standard-Prozedur WRITE (entspricht BDOS-Funktion 2) Verfälschungen auftraten. Die Gründe dafür (Eigenheit des Betriebssystems oder des Turbo-Laufzeitmoduls) habe ich jedoch nicht näher untersucht. Aus diesem Grund erfolgt die Ausgabe aller Zeichen mit einem Code oberhalb 128 mittels der BIOS-Funktion 4 (CONOUT).
Da der anzulegende Grafik-Bildspeicher die Hälfte des für Variablen zur Verfügung stehenden Speicherplatzes einnimmt, kann er bei Bedarf in den Heap verlagert werden. Aus Gründen der Übersichtlichkeit wurde hier darauf verzichtet.
Soll nur ein Teil des Bildschirms für die Grafik genutzt werden, so können die Werte für MX und MY geändert werden, wobei die Anzahl der Pixel in X-Richtung stets ein Vielfaches von 8 und in Y-Richtung ein Vielfaches von 16 sein müssen. Die Werte in der Typdeklaration für GRAFIK_SCREEN können dann auch verringert werden, um Speicherplatz zu sparen. Die beiden oben genannten Prozeduren müssen dabei nicht verändert werden.
Beim Druck wird nur der Inhalt des Grafikspeichers ausgegeben. Eventuell auf dem Bildschirm vorhandener Text wird nicht mit gedruckt. Es handelt sich also nicht um eine echte Hardcopy. Soll Text auch gedruckt werden, so ist er pixelweise im Grafikspeicher aufzubauen.
Für den Druck wurde der Grafikmodus 4 (Bildschirmgrafik) des Druckers verwendet. Über diese Funktion verfügen auch andere Drucker (z. B. LX 86). Das ausgegebene Bild ist damit genau 80 Zeichen breit. Dadurch entsteht aber eine leichte Verzerrung (Stauchung in X-Richtung). Die Verwendung des Grafik-Modus 5 (Plotter 1:1) gibt die Grafik verzerrungsfrei aus, sie steht aber nicht auf allen Druckern zur Verfügung und erfordert breiteres Papier.

Geräte ohne Grafik-Karte
Nachfolgende Bemerkungen sollen als Anregung verstanden werden, da sie mangels Gerätetechnik nur auf einem A 7100 mit Grafik-Karte simuliert werden konnten.
Auf Geräten ohne Grafik-Karte kann das beschriebene Verfahren nicht angewendet werden, da hier der BWS zeichenweise aufgebaut ist und damit die Pixelspeicher-Funktion entfällt.

```
  1: 0 PROGRAM grafik;
  2: 0 {*************************************************************}
  3: 0 {* Einzelpunktgrafik fuer den A7100 mit KGS                  *}
  4: 0 {* Bernd Matzke, Rosa-Luxemburg-Strasse 6, Delitzsch 7270    *}
  5: 0 {*************************************************************}
  6: 0 CONST xmax = 639;  {Pixelzahl x-Richtung -1; xmax:=n* 8 -1, n=1,2..79}
  7: 0       ymax = 399;  {Pixelzahl y-Richtung -1; ymax:=n*16 -1, n=1,2..25}
  8: 0       Pi   = 3.141;
  9: 0       video_kaus = ^['[?14l';                         {kursor aus}
 10: 0       video_kein = ^['[?14h';                         {kursor ein}
 11: 0       video_wrap = ^['[?7h';                          {rollen ein}
 12: 0       video_nowr = ^['[?7l';                          {rollen aus}
 13: 0 TYPE  zeile = ARRAY[0..79] OF BYTE;
 14: 0       grafik_screen = ARRAY[0..ymax] OF zeile;
 15: 0 VAR   grs:grafik_screen;
 16: 0
 17: 0 PROCEDURE conout(zeichen:BYTE);      {aufruf bios-funktion 4 (conout)}
 18: 0 TYPE cpu_reg = RECORD                                   {CPU-Register}
 19: 0                  ax,bx,cx,dx,bp,si,di,ds,es,flags:INTEGER;
 20: 0                END;
 21: 0      block   = RECORD                        {block mit uebergabewerten}
 22: 0                  nr,cl,ch,dl,dh:BYTE;
 23: 0                END;
 24: 0 VAR  register: cpu_reg;
 25: 0      bios    : block;
 26: 0
 27: 0 BEGIN  {conout}
 28: 1   bios.nr:=4;          {bios-funktionsnummer, direkte consolenausgabe}
 29: 1   bios.cl:=zeichen;
 30: 1   register.cx:=$32;              {nummer der bdos-funktion fuer bios-ruf}
 31: 1   register.ds:=Seg(bios);          {segmentadresse bios-parameter-block}
 32: 1   register.dx:=Ofs(bios);           {offsetadresse bios-parameter-block}
 33: 1   Bdos(register);
 34: 1 END;
 35: 0
 36: 0 PROCEDURE pixel(x,y,modus:INTEGER);            {modifizieren eines pixels}
 37: 0 {modus  0: loeschen                                                    }
 38: 0 {       1: setzen                                                      }
 39: 0 {       2: invertieren                                                 }
 40: 0 VAR alpha_spalte:INTEGER;
 41: 0     yy,                   {erste pixelzeile des zu aktivierenden feldes}
 42: 0     i:INTEGER;
 43: 0
 44: 0 BEGIN  {pixel}
 45: 1   IF (x>=0) AND (x<=xmax) AND (y>=0) AND (y<=ymax) THEN BEGIN
 46: 2     IF NOT modus IN [0..2] THEN modus:=1;
 47: 2     y:= ymax-y;                                     {ursprung links unten}
 48: 2     alpha_spalte:= x DIV 8;
 49: 2     yy:=y DIV 16 * 16;
 50: 2     CASE modus OF
 51: 3       0:grs[y,alpha_spalte]:=grs[y,alpha_spalte] AND
 52: 3            NOT(1 SHL (7 - x MOD 8));
 53: 3       1:grs[y,alpha_spalte]:=grs[y,alpha_spalte] OR
 54: 3            1 SHL (7 - x MOD 8);
 55: 3       2:grs[y,alpha_spalte]:=grs[y,alpha_spalte] XOR
 56: 3            1 SHL (7 - x MOD 8);
 57: 3     END;
 58: 2     conout(16); conout(0);           {zeichen mit nummer 128 definieren}

 59: 2     FOR i:=0 TO 15 DO conout(grs[yy+i,alpha_spalte]);
 60: 2     Write(video_nowr);                                {rollen abschalten}
 61: 2     GotoXY(1+alpha_spalte,1+y DIV 16);
 62: 2     conout(128);                                     {zeichen auf schirm}
 63: 2     Write(video_wrap);
 64: 2   END;
 65: 1 END;
 66: 0
 67: 0 PROCEDURE print_grafik;
 68: 0 VAR ascii_spalte,
 69: 0     drucker_zeile,                         {zeilennummer der druckzeile}
 70: 0     pixel_spalte,                      {bitnummer innerhalb eines bytes}
 71: 0     nadel,                              {nummer der nadel am druckkopf}
 72: 0     p,                                                 {hilfsvariable}
 73: 0     s:INTEGER;                                {zu sendendes grafik-byte}
 74: 0
 75: 0 BEGIN {print_grafik}
 76: 1   Write(LST,#27,#51,#24);        {zeilenabstand fuer lueckenlosen druck}
 77: 1   FOR drucker_zeile:= 0 TO ymax DIV 8  DO           {immer 8 pixelzeilen}
 78: 1   BEGIN                        {grafik-modus fuer eine druckzeile einschalten}
 79: 2     Write(LST,#27,#42,#4,Chr((xmax+1) MOD 256),Chr((xmax+1) DIV 256));
 80: 2       {bildschirmgrafik = hardcopy = 80 punkte/zoll,                    }
 81: 2       {bildbreite ist 8 zoll bzw. 20 cm                                 }
 82: 2 {  Write(LST,#27,#42,#5,Chr((xmax+1) MOD 256),Chr((xmax+1) DIV 256));}
 83: 2 {  plottergrafik mit seitenverhaeltnis 1:1 72 punkte/zoll             }
 84: 2 {  bildbreite rund 9 zoll = 22,5 cm                                   }
 85: 2     FOR ascii_spalte:=0 TO xmax DIV 8 DO BEGIN
 86: 3                                              {alle zeichen einer zeile}
 87: 3       FOR pixel_spalte:=7 DOWNTO 0 DO  BEGIN
 88: 4                                     {alle pixelspalten eines zeichens}
 89: 4         s:=0;
 90: 4         p:=1;
 91: 4         FOR nadel:=7 DOWNTO 0 DO BEGIN                {fuer alle 8 zeilen}
 92: 5           s:= s+ p* Ord(Odd(grs[8 *drucker_zeile +nadel, ascii_spalte]
 93: 5                          SHR pixel_spalte));
 94: 5           p:=p*2;
 95: 5         END;
 96: 4         Write(LST,Chr(s));
 97: 4       END;
 98: 3     END;
 99: 2     WriteLn(LST);
100: 2   END;
101: 1   Write(LST,#27,#64);                                     {drucker-reset}
102: 1 END;
103: 0
104: 0 PROCEDURE bild;
105: 0 VAR dx,dy:REAL;
106: 0     n,i,k,l,zx,zy:INTEGER;
107: 0 BEGIN
108: 1   Write(video_kaus);
109: 1   GotoXY(15,3);
110: 1   Write(' Demonstrationsprogramm    Einzelpunktgrafik ');
111: 1   zx:=xmax DIV 2;
112: 1   zy:=ymax DIV 2;
113: 1   FOR n:=0 TO xmax DO BEGIN                     {rahmen, koordinatenkreuz}
114: 2     pixel(n,0,1); pixel(n,zy,1); pixel(n,ymax,1);
115: 2   END;

116: 1   FOR n:=0 TO ymax DO BEGIN
117: 2     pixel(0,n,1); pixel(zx,n,1); pixel(xmax,n,1);
118: 2   END;
119: 1   dx:=8*Pi/xmax;
120: 1   dy:=ymax/xmax;
121: 1   FOR n:=0 TO xmax DO                                      {cosinuskurve}
122: 1     pixel(n,zy + Round(dy * (zx-Abs(zx-n)) * Cos(-4*Pi+n*dx) ),1);
123: 1   FOR n:=0 TO Round(0.71*zy) DO BEGIN                              {kreis}
124: 2     i:=Round(Sqrt(1.0 * zy*zy -(n*n)) );
125: 2     pixel(zx-n,zy+i,1); pixel(zx-n,zy-i,1);
126: 2     pixel(zx-i,zy+n,1); pixel(zx-i,zy-n,1);
127: 2     pixel(zx+n,zy+i,1); pixel(zx+n,zy-i,1);
128: 2     pixel(zx+i,zy+n,1); pixel(zx+i,zy-n,1);
129: 2   END;
130: 1   k:=xmax DIV 10;
131: 1   l:=ymax DIV 10;
132: 1   FOR n:=k TO 2*k DO
133: 1     FOR i:=l TO 2*l DO
134: 1       pixel(n,i,1);                              {volles rechteck zeichnen}
```

```
135: 1   FOR n:=Trunc(1.3*k) TO Trunc(1.7*k) DO
136: 1     FOR i:=Trunc(1.3*l) TO Trunc(1.7*l) DO
137: 1       pixel(n,i,0);                          {rechteckflaeche loeschen}
138: 1   FOR n:=Trunc(0.8*k) TO Trunc(2.2*k) DO
139: 1     FOR i:=Trunc(1.4*l) TO Trunc(1.6*l) DO
140: 1       pixel(n,i,2);                          {demonstration xor}
141: 1   Write(video_kein);
142: 1 END;
143: 0
144: 0 BEGIN
145: 1   ClrScr;
146: 1   FillChar(grs,SizeOf(grs),0);
147: 1   bild;
148: 1   print_grafik;
149: 1 END.
```

```
 1: 0 PROGRAM grafik2;
 2: 0 {ANREGUNG einzelpunktgrafik fuer A7100 = mit ABS }
 3: 0 CONST xmax = 127;
 4: 0       ymax = 127;
 5: 0       Pi   = 3.141;
 6: 0       video_kaus = ^['[?14l';                         {kursor aus}
 7: 0       video_kein = ^['[?14h';                         {kursor ein}
 8: 0       video_wrap = ^['[?7h';                          {rollen ein}
 9: 0       video_nowr = ^['[?7l';                          {rollen aus}
10: 0 TYPE  zeile = ARRAY[0..15] OF BYTE;
11: 0       grafik_screen = ARRAY[0..ymax] OF zeile;
12: 0 VAR   grs:grafik_screen;
13: 0       i,n:INTEGER;
14: 0
15: 0 PROCEDURE conout(zeichen:BYTE);
16: 0 BEGIN
17: 1 {siehe programm grafik}
18: 1 END;
19: 0
20: 0 PROCEDURE pixel(x,y,modus:INTEGER);          {modifizieren eines pixels}
21: 0 {modus  0: loeschen                                                  }
22: 0 {       1: setzen                                                    }
23: 0 {       2: invertieren                                               }
24: 0 VAR xx,yy,i,zeichen_nr:INTEGER;
25: 0
26: 0 BEGIN  {pixel}
27: 1   IF (x>=0) AND (x<=xmax) AND (y>=0) AND (y<=ymax) THEN BEGIN
28: 2     IF NOT modus IN [0..2] THEN modus:=1;
29: 2     y:= ymax-y;                                 {ursprung links unten}
30: 2     xx:= x DIV 8;
31: 2     yy:=y DIV 16 * 16;
32: 2     zeichen_nr:=xx+yy;
33: 2     CASE modus OF
34: 3       0:grs[y,xx]:=grs[y,xx] AND NOT(1 SHL (7 - x MOD 8));
35: 3       1:grs[y,xx]:=grs[y,xx] OR 1 SHL (7 - x MOD 8);
36: 3       2:grs[y,xx]:=grs[y,xx] XOR 1 SHL (7 - x MOD 8);
37: 3     END;
38: 2     conout(16); conout(zeichen_nr);             {zeichen definieren}
39: 2     FOR i:=0 TO 15 DO conout(grs[yy+i,xx]);
40: 2     Write(video_nowr);                          {rollen abschalten}
41: 2     GotoXY(1+xx,1+y DIV 16);
42: 2     conout(128 + zeichen_nr);                   {zeichen auf schirm}
43: 2     Write(video_wrap);
44: 2   END;
45: 1 END;
46: 0
47: 0 PROCEDURE bild;
48: 0 VAR dx:REAL;
49: 0     n:INTEGER;
50: 0 BEGIN
51: 1   Write(video_kaus);
52: 1   FOR n:=0 TO xmax DO BEGIN                  {rahmen, koordinatenkreuz}
53: 2     pixel(n,0,1); pixel(n,ymax DIV 2,1); pixel(n,ymax,1);
54: 2   END;
55: 1   FOR n:=0 TO ymax DO BEGIN
56: 2     pixel(0,n,1); pixel(xmax DIV 2,n,1); pixel(xmax,n,1);
57: 2   END;
58: 1   dx:=8*Pi/xmax;
59: 1   FOR n:=0 TO xmax DO
60: 1     pixel(n,63 + Round((63 - Abs(63-n)) * Cos(-4*Pi+n*dx)),1);
61: 1   Write(video_kein);
62: 1 END;
63: 0
64: 0 BEGIN
65: 1   ClrScr;
66: 1   FillChar(grs,SizeOf(grs),0);
67: 1   FOR n:=0 TO 127 DO BEGIN
68: 2     conout(16);
69: 2     conout(n);
70: 2     FOR i:=0 TO 15 DO conout(0); {frei definierbare zeichen loeschen}
71: 2   END;
72: 1   bild;
73: 1   GotoXY(1,15);
74: 1   FOR n:=0 TO 7 DO BEGIN                {nochmalige ausgabe zur kontrolle}
75: 2     FOR i:=0 TO 15 DO
76: 2       conout(128+n*16+i);
77: 2     WriteLn;
78: 2   END;
79: 1 END.
```

Verzichtet man aber auf die anderweitige Nutzung des frei programmierbaren Zeichengenerators, kann mit dessen Hilfe eine Einzelpunktgrafik mit einer Auflösung von 128 × 128 Punkten aufgebaut werden, was für bescheidene Anwendungen unter Umständen ausreicht. Dabei besteht das Grafik-Feld aus den 128 frei definierbaren Zeichen, die in 8 Zeilen zu 16 Spalten dargestellt werden.

Im Demonstrationsprogramm wird zuerst eine einfache Grafik aufgebaut, anschließend wird der Inhalt des frei programmierbaren Zeichengenerators nochmals ausgegeben, um seinen Inhalt zu kontrollieren.

Es wäre denkbar, die Grafik wie im ersten Beispiel mit hoher Auflösung zu berechnen, zu speichern und bei Bedarf zu drucken, aber zu Kontrollzwecken mit der verringerten Auflösung auf dem Monitor anzuzeigen. Damit sind auch auf einem nicht grafikfähigen A 7100 bescheidene Grafikanwendungen möglich.

KC 85/1 generiert 24 × 80 Zeichen

Das Bildschirmformat 24 × 80 hat sich als Standard in der Computertechnik durchgesetzt. Es ermöglicht die Anwendung von leistungsfähigen Textverarbeitungsprogrammen (zum Beispiel Wordstar). Diejenigen, die ihren Homecomputer aufwerten und gegebenenfalls mit einer RAM-Floppy oder einem Diskettenlaufwerk erweitern wollen, können mit geringstem Hardwareaufwand den Schritt zum kompatiblen CPM-Rechner vollziehen. Um der „alten 40-Zeichen-Software" und vor allem den Spielen nicht den Garaus zu machen, ist eine Umschaltung zwischen 40- und 80-Zeichen-Modus wünschenswert.

Da der KC 85/1 bereits ein Bildschirmformat von 24 × 40 Zeichen besitzt, ist der Umbau denkbar einfach. Benötigt werden lediglich vier zusätzliche Schaltkreise und natürlich zwei oder für die Farbvariante vier zusätzliche Bildwiederholspeicher-RAMs, die nach alter Bastlermanier huckepack auf die bereits im Gerät vorhandenen Schaltkreise (nach Ablauf der Garantiezeit!) aufgelötet werden können.

Damit in der gleichen Zeiteinheit die doppelte Anzahl von Zeichen aus dem Bildwiederholspeicher ausgelesen werden kann, ist als erstes eine Verdopplung der Grundtaktfrequenz erforderlich. Das ist kein Problem, da der orginal vom CPU-Grundtakt synchronisierte Pixeltaktgenerator durch den Einsatz eines 14,745-MHz-Standardquarzes entsprechend modifiziert werden kann. Nach dieser Änderung synchronisiert natürlich kein Fernseher mehr. Um das alte Signalspiel und vor allem die Zeitverhältnisse des Zeilen- sowie des Bildsynchronsignals wieder zu restaurieren, muß als nächstes das Teiler-Flipflop U3A noch vor den ersten Zähler (D50) geschaltet werden, und alles ist wieder beim alten. Nach diesen vielleicht etwas sinnlos erscheinenden Vorarbeiten kommt nun der eigentliche Trick. Mit Hilfe des Multiplexers U1 (DL257) wird nicht nur der Pixelgrundtakt CLK-FAR des Bildwiederholspeichers zwischen Normal- und Doppeltakt umgeschaltet, sondern auch ein Eingang (PIN 9) des Gatters D27C, welches das Zeilensynchronsignal dadurch auch im 80-Zeichen-Modus richtig generiert. Und das war's dann auch schon fast. Der Ausgang QC des Zählers D50 dient im 80-Zeichen-Modus der Selektierung des zusätzlichen Bildwiederholspeichers und – falls vorhanden – Farbattributspeichers. Von Vorteil ist, daß nach wie vor nur der Adreßbereich von E800H bis EFBFH für den Bildwiederholspeicher benötigt wird. Damit der Zusatzspeicher auch vom Rechner beschrieben werden kann, müssen entsprechend dem Originalschaltplan die Auswahlsignale CS 1 und CS 2 für beide Speicherebenen mit Hilfe eines DL 051 getort werden. Da im Originalgerät der CS-Eingang der Bildwiederholspeicherschaltkreise U214 auf Masse liegt, darf das Auftrennen dieser Verbindung natürlich nicht vergessen werden.

Bevor sich kurze Erläuterungen zur Software anschließen, noch ein Wort zur Umschaltlogik. Verwendet werden zwei Flipflops, die über den bereits vorhandenen Adreßdekoder D8 getaktet werden und den Pegel des Adreßsignals AB0 übernehmen. Auf diese Weise lassen sich die Flipflops mit einfachen OUT-Befehlen schalten, wobei das eine (U4A) zwischen dem 80- und dem 40-Zeichen-Modus umschaltet und das andere (U4B) für die CPU-seitige Adressierung des Bildspeichers im 80-Zeichen-Modus benutzt wird.

Soweit die Hardware, nun zur Software. Der aufmerksame Leser hat sicherlich bereits festgestellt, daß nunmehr im 80-Zeichen-Modus auf jeder Bildschirm-Speicheradresse zwei Zeichen eingeschrieben werden müssen, die auf dem Bildschirm auf benachbarten Positionen erscheinen. Dies erklärt sich aus der Tatsache, daß unser zusätzliches Adreß- bzw. Selektierungsbit (Ausgang QC von D50) mit der doppelten Taktfrequenz gegenüber dem Adreßsignal A0 läuft. Die Tafel 1 verdeutlicht die Zeichendarstellung und die Adressierung im 40- und im 80-Zeichen-Modus (ZM).

Man erkennt, daß im 80-Zeichen-Modus alle Zeichen, die auf einer ungeraden Bildschirmposition dargestellt werden sollen, in den Zusatzbildwiederholspeicher geschrieben werden müssen. Gleiches gilt natürlich auch für den Farbattributspeicher.

Die kurze Befehlsfolge in Bild 1, die in den 80-Zeichen-CRT-Treiber entsprechend eingebaut werden muß, ermöglicht die kursorpositionsabhängige Steuerung des Auswahl-Flipflops.

Zum Schluß noch einige Hinweise zum Schaltplan (Bild 2). Alle Leitungen, die mit einem ∗ gekennzeichnet sind, sowie die Bezeichnung der Bauelemente entsprechen dem Origi-

Tafel 1 Zeichendarstellung auf dem Bildschirm

Bildspeicheradresse	EC00H	EC01H	EC02H	EC03H	EC04H	EC05H ...
40-ZM	Z0	Z1	Z2	Z3	Z4	Z5
80-ZM	Z0 Z1	Z2 Z3	Z4 Z5	Z6 Z7	Z8 Z9	Z10 Z11 ...

```
        ;in E : auszugebendes Zeichen
        ;in HL: absolute Kursoradresse

C080:   PUSH    HL              ;absolute Position retten
        SRL     H               ;BS-Pos.= abs. Pos. / 2
        RR      L
        JR      NC,gerade       ;Test: gerade/ ungerade
        OUT     (0A1H),A        ;ungerade: Auswahl-FF setzen
gerade: LD      A,0ECH          ;BS-Adresse berechnen:
        ADD     A,H             ;BS-offset EC00h addieren
        LD      H,A
        LD      (HL),E          ;Zeichen ausgeben
        OUT     (0A0H),A        ;Auswahl-FF rücksetzen
        POP     HL
        RET
```

Bild 1

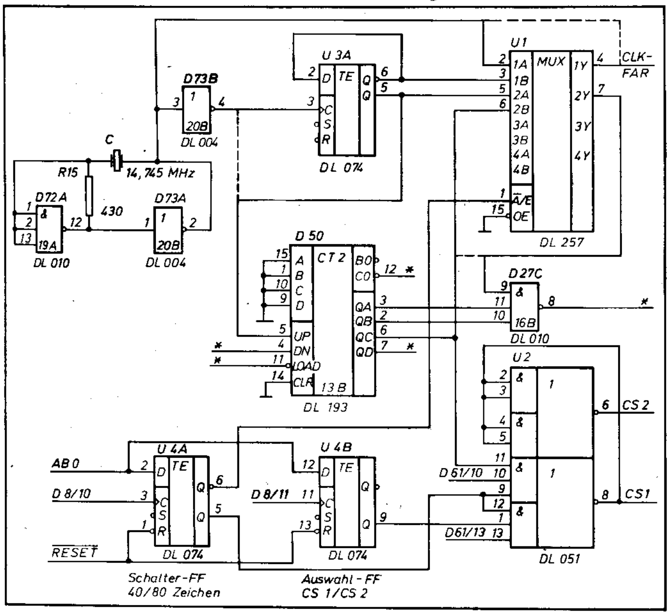


Bild 2
Out (0A8H) oder Reset ---→ 40-Zeichen-Modus
Out (0A9H) ---→ 80-Zeichen-Modus
Out (0A0H) ---→ CS 1
Out (0A1H) ---→ CS 2

nalschaltplan /1/ und wurden nicht verändert. Die auf der Leiterplatte zu trennenden Leiterzüge sind gestrichelt gezeichnet. Vor dem Nachbau der angegebenen Schaltung sollte man den Originalschaltplan gründlich studieren.
Abschließend noch ein Wort zum Fernseher. Aufgrund der hohen Pixelfrequenz von 14,7 MHz ist auf jeden Fall ein Fernsehgerät mit BAS-Eingang (gegebenenfalls das Gerät nachrüsten) erforderlich. Da im 80-Zeichen-Modus die horizontale Ausdehnung der Zeichen halbiert wird, ist die Benutzung des IBM-Zeichensatzes /2/ (außer den Grafikzeichen ab 80H) im Interesse einer guten Lesbarkeit anzuraten.

Literatur

/1/ Stromlaufpläne des Kleincomputers robotron Z 9001. Firmenschrift des VEB Robotron-Messelektronik „Otto Schön" Dresden 1986
/2/ Bauer, Th.: 8 × 8 – ein Font für alle Fälle. Mikroprozessortechnik, Berlin 3 (1989) 2 S. 53

Christian Schiewe

Hinweis zum Beitrag

„Temperaturmessung mit dem KC 85/1"

MP 1/1990, Seite 26

Im genannten Artikel wird die Ankopplung eines Digitalthermometers DTM 2010 unserer Produktion an einen Kleincomputer KC 85/1 beschrieben. Dabei ist nach Aussagen des Thermometerwerkes folgendes zu beachten.

- Die Wertigkeit der Ausgangsleistungen beträgt:
DIO1: 2^0 ... DIO8: 2^8
- Die aktive Flanke für das Signal NRFD ist ebenfalls L-H.
- Die Verbindung NRFD und NDAC (s. Schaltbild) wird bei der gegenwärtig produzierten Gerätekonzeption DTM 2100 nicht mehr zum Erfolg führen, denn NDAC als Quittungssignal muß erst nach erfolgter Übernahme aktiv gesetzt werden.

Aus unserer Sicht müßte die vorgestellte Lösung in der Anwendung der Maschinenroutine bestehen, um den Dreidraht-Handshake problemlos ausführen bzw. den Normalbetrieb des DTM ebenfalls auswerten zu können.

⊠ KONTAKT ☏

Thermometerwerk Geraberg, Herr Trommer, Elgersburger Straße 1, Geraberg, 6306; Tel. Ilmenau 62 61 – 62 65.

Sicheres Entriegeln von Diskettenlaufwerken

Die am A 7100 eingesetzten Diskettenlaufwerke lassen sich nach häufiger Benutzung oftmals nicht mehr ohne weiteres entriegeln. Obwohl der Hebel freigegeben wird, lassen sich die Disketten nicht entnehmen. Viele Nutzer versuchen, mit unsanften Mitteln das Entriegeln zu erzwingen, was zu Schäden an Diskette und Laufwerk führen kann.
Durch einen einfachen Zusatz im Diskettenlaufwerk kann die ordnungsgemäße Funktion dauerhaft gewährleistet werden. Dazu wird auf die bewegliche Kopfplatte eine Schiene aufgeschraubt (siehe Skizze), die vom Riegel beim Öffnen zwangsweise mitgeführt wird. Ein Loch mit Gewinde M3 ist auf der Kopfplatte bereits vorhanden.
Zu beachten ist, daß die zusätzliche Schiene die Länge von 72 mm zwischen Mitte Bohrung und vorderem Ende nicht überschreiten darf. Es wird darauf aufmerksam gemacht, daß eventuelle Garantieansprüche nach dem Eingriff erlöschen.

⊠ KONTAKT ☏

Technische Universität Dresden, Sektion Informationstechnik, Bereich Kommunikations- und Computertechnik, Mommsenstraße 13, Dresden, 8027; Tel. 4 63 39 41

Thomas Johannsen

MS-DOS-Tip

Tastaturtreiber-Patch

Haben Sie auch das Problem, daß Sie auf Ihrem EC 1834 im Numerikblock der Tastatur K 7673.02 die Komma-Taste mit dem Dezimalpunkt belegen wollen?
Dann kann ich Ihnen einen kleinen Patch im Tastaturtreiber Keybgd.com (Version 1.0 vom 01.01. 1988) vorschlagen. Ändern Sie das unter der Absolutadresse 0000:075F stehende Byte **2C** in **2E**. Wenn Sie mit Mdisk arbeiten, finden Sie das Byte im relativen Sektor 0000003 Adresse 0351 (015FH).

Uwe Hampel

Fehler in Turbo-Pascal 5.0

Bei der Verwendung von Turbo-Pascal 5.0 unter MS-DOS mit mathematischem Koprozessor (i8087) kommt es mitunter zu undefinierten Programmzuständen, bei denen auch die Fehlersuche mit Hilfe der integrierten Entwicklungsumgebung erfolglos bleibt. Das Programm bleibt „hängen". So läuft zum Beispiel folgender Programmteil nicht korrekt (der Fehler tritt bei der Berechnung des Sinus auf):

```
        {$E-,F-,N+,O-,R-,S+}
var s1,s2:string[5];
        x,y:extended;
begin
        move(s1,s2,6)
        x:=0.5; y:=sin(x);
        writeln(y); readln;
end.
```

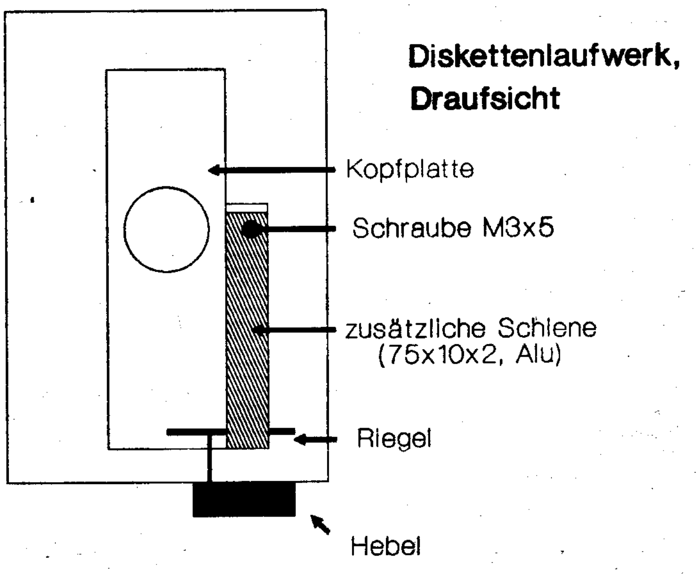
Schreibt man hingegen **move(s2,s1,6)**, tritt der erwähnte Effekt nicht auf. Die Ursache ist das nicht verwaltete Direction-Flag, welches ein ständiges Aufrufen des Interrupts 3EH (für die Ansteuerung des Koprozessors verwendet) durch Turbo-Pascal bewirkt, da die Rücksprungadresse auf dem Stack falsch manipuliert wird. Genauen Aufschluß erhält man durch eine Verfolgung des Interrupts auf Assemblerebene mit dem Turbo-Debugger. Abhilfe schafft das in Bild 1 dargestellte Unit, welches den Interrupt entsprechend verändert. Das Unit muß nur im Hauptprogramm (nicht in anderen Units) eingefügt werden.

Jens Jacobi

```
{$A+,B-,D+,E-,I+,L+,N+,O-,R-,S+,V-}
unit co87;
interface
     uses dos;
implementation
     var org3e, orgexit :pointer;
{$F+}
procedure cocorect;
 begin
  inline(
   $5D/             { pop bp          }
   $fc/             { cld             }
   $ff/$2e/org3e { jmp far [ds:org3e] }
         );  end;

procedure reset;
     begin setintvec($3e,org3e);
           exitproc:= orgexit;
     end;
begin
  getintvec($3e,org3e);
  setintvec($3e,@cocorect);
  orgexit:=exitproc;
  exitproc:=@reset;
end.
```

Bild 1 Unit co87



Erweiterungskarte P8000-Terminal

Über eine Adaption der Erweiterungskarte auf dem P8000-Terminal-Grundgerät und eine Neuimplementierung der Firmware werden zusätzlich folgende Eigenschaften für das P8000-Terminal wirksam: 80 Zeichen in 24 oder 25 Zeilen, drei durch Tastatur oder Steuerfolgen umschaltbare Zeichensätze, Pseudografik-Fähigkeit, über Steuerfolgen umschaltbare Terminal-Modi, WS-kompatible Kursor- und Funktionstasten, Kursor transparent/nontransparent, Tastatur on/off, Hell-/Dunkeltastung Monitor, Hardcopy- und Print-Modus.
An die auf der Erweiterungskarte vorhandene serielle Schnittstelle kann ein Drucker angeschlossen werden. Damit kann das P8000-Terminal wechselseitig als Steuergerät, als Druckeinheit oder als Hardcopyeinheit verwendet werden.

Zentrum für Forschung und Technologie, Kombinat VEB Elektro-Apparate-Werke Berlin-Treptow, Storkower Straße 101, Berlin, 1055; Tel. 4 38 85 32/4 38 85 69
Kühle/Schmidt

Konvertierungsprogramm WORDTP

Durch die schrittweise Ausrüstung des KC 85/3 und /4 mit Diskettenlaufwerken wird bei vielen KC-Nutzern der Wunsch entstehen, das Textverarbeitungssystem Wordpro gegen das komfortablere TP von Sömmerda auszutauschen. Um die in den letzten Jahren entstandenen Wordpro-Textkonserven nicht neu mit dem Textprozessor eintippen zu müssen, ist von uns ein Konvertierungsprogramm in Turbo-Pascal geschrieben worden, das Wordprotexte automatisch in .TXT-Dateien umwandelt. Dabei werden alle Steuerzeichen mit konvertiert, so daß das Druckbild des ursprünglichen Textes auf dem KC mit Floppy oder PC 1715 oder A 5120/30 genau wie das Wordpro-Druckbild aussieht. Das Programm WORDTP kann bei uns bei Zusendung einer Diskette und eines formlosen Auftrages zum Preis von 300,– Mark erworben werden. Bei Bedarf wird das TPL kostenlos mitgeliefert.

Zwischengenossenschaftliche Einrichtung Landbauprojektierung Zwickau, Abt. Rechentechnik, PSF 802, Reichenbacher Straße 158, Zwickau, 9502; Tel. 78 82
Uhlig

Druckprogramm mit Syntaxprüfung

Es wurde ein Turbo-Pascal-Programm entwickelt, das den Ausdruck von REDABAS-, dBase- und Clipper-Quelltexten wirkungsvoll unterstützt. Vor dem Ausdrucken wird ein elementarer Syntaxtest nach den Strukturanweisungen (DO WHILE, DO CASE, IF und FOR) durchgeführt. Hierbei auftretende Strukturfehler können mit einem integrierten Editor bearbeitet werden. Der korrekte Quelltext wird durch Eindrücken formatiert und durch Zeilennummer und Schachtelungstiefe ergänzt auf einem Drucker ausgegeben. Der Nutzer wird durch On-line-Hilfestellung im Programm unterstützt. Benötigt werden ein IBM-kompatibler PC, ein Epson-kompatibler Drucker und MS-DOS 3.10 bis 3.30. Die Nachnutzungsgebühr beträgt 350 M.

VEB Automobilwerk Eisenach, Abt. TVQ, Rennbahn 8, Eisenach, 5900; Tel. 5 95 67
Dr. Remmler

DELALL

Das Programm DELALL kann Dateien gleichen Namens oder Typs vom gesamten Datenträger löschen. Jedes der angelegten Verzeichnisse auf dem aktuellen Datenträger wird nach der entsprechenden Datei durchsucht. Das Zeichen * im Dateinamen ersetzt ein oder mehrere Zeichen im Dateinamen. Durch die Eingabe eines Parameters kann der Nutzer entscheiden, ob eine Löschung automatisch erfolgen soll oder nicht. Auf diesem Wege kann der Nutzer relativ schnell seinen Datenträger bereinigen und freien Speicherplatz schaffen. Diese Lösung ist für alle A 7150, EC 1834 ... nutzbar. Geliefert werden die Programmdateien und eine Anwenderbeschreibung auf nutzereigenen 5 1/4-Zoll-Disketten (720 KByte).

VEB Datenverarbeitungszentrum Neubrandenburg, Bereich Absatz und Beschaffung, Straße der Befreiung 12, PSF 20, Neubrandenburg, 2000; Tel. 58 73 33
Claus

Echtzeitsystem lasFORTH

Das Forth-Echtzeitsystem lasFORTH wurde für die Implementierung von Multitasksystemen für eingebettete Rechnersysteme der Prozessorfamilien Z 80 und 8086 entwickelt. lasFORTH enthält selbst keinen Übersetzer. Zur Programmentwicklung können das comFORTH-Basispaket (off-line) und das System comFORTH-plus (on-line) eingesetzt werden. Die Generierung erfolgt mit Hilfe des CrossCompilers. lasFORTH wurde gegenüber gewöhnlichen Forth-Systemen um eine Prozeßverwaltungsschicht erweitert.
Die Definition von Tasks und Synchronisationsobjekten ist bereits bei der Systemgenerierung bzw. unter comFORTH-plus möglich.

FORTecH Software, Albert-Einstein-Straße 2, Rostock 6, 2500; Tel. 40 52 02
Dr. Pfüller

NETBASE

Das Datenbanksystem NETBASE erlaubt die Arbeit mit Datenbankfiles im lokalen Rechnernetz SCOM-LAN. In der ersten Ausbaustufe erreicht das System folgende Leistungsparameter: simultane Arbeit von 8- und 16-Bit-Rechnern, interaktive Nutzerumgebung, Datenbestände verbleiben im REDABAS-Format, Dateizugriffs- und Bearbeitungszeiten entsprechen auf allen Stationen internationalen Normen, automatisches Record- und Filelocking, Filelocking nur beim physischen Löschen von Datensätzen sowie beim Indizieren und Sortieren der Datei, Arbeit in mehreren Datenbankebenen.

VEB Applikationszentrum Elektronik Berlin, IfAM Berlin, PF 211, Berlin, 1035; Tel. 5 88 38 41
Raschik

Steckeinheit für die Bildverarbeitung

Die entwickelte kombinierte Bildaufnahme-/Bildwiedergabebaugruppe realisiert ein Bildraster von 256 × 256 Pixel bei einer Auflösung von 16 Grau- bzw. Farbstufen. Kernstück der Steckeinheit ist der GDC U 82720 D. Dieser erlaubt die Nutzung sowohl als Grafiksteckeinheit mit den unterstützenden Funktionen des GDCs als auch durch ein vorhandenes Videointerface die Eingabe von Kamerabildinformationen. Die Zoom- und Pan-Funktionen des GDCs unterstützen die Bildverarbeitung von verschiedenen Rasterbildern und Rasterbildgrößen und von Bildfolgen.
Der Einsatz der Steckeinheit ist in den Rechnern K 1520 und EC 1834 möglich. Die Lösung liegt als technische Dokumentation und in begrenztem Umfang als fertige Leiterkarte vor.

VEB Chemieanlagenbaukombinat Leipzig-Grimma, Stammbetrieb, HA 422000, Bahnhofstraße 3–5, Grimma, 7240; Tel. 63/36 83
Dr. Weigel

Speichererweiterung von Robotron-Leiterkarten K 3820

Zur Nachnutzung wird eine Umbauvorschrift angeboten, die es ermöglicht, die bisher im Handel befindlichen Robotron-Festwertspeicherleiterkarten K 3820 (16 KByte) auf eine maximale Speicherkapazität von 64 KByte aufzustocken. Des weiteren wird eine Modifizierungsvariante für den gemischten Einsatz von EPROMs und RAMs angeboten.

Technische Universität Dresden, Sektion Informationstechnik (09), Bereich 2, Mommsenstraße 13, Dresden, 8027; Tel. 4 63 22 86
Nicklisch

Rationelles Arbeiten mit AutoCAD

Für die Optimierung Ihrer Arbeit mit dem Programmpaket AutoCAD bieten wir Ihnen folgende Softwareprodukte:
SETUP ermöglicht ein bildschirmmenügestütztes Einrichten Ihrer Zeichnungsblätter, einschließlich Faltmarkierungen, Schriftfeldaufbau und Layerorganisation.
QuickKey steuert die Mehrfachbelegung der Funktionstasten mit Auto-Cad-Funktionen, Befehlen oder Befehlsmacros.
TextTrans bietet die Möglichkeit, ASCII-Files in oder aus Bild-(.dwg-)Dateien ein- oder auszulesen.
TR ist ein in AutoCAD integrierter Taschenrechner mit 14 mathematischen Funktionen, einem rechnenden Speicher und speziellen Funktionen zur Längen- und Winkelberechnung.
Alle Programme sind auf IBM PC/XT, /AT oder Kompatiblen mit mindestens 640 KByte RAM unter MS-DOS Version 3.xx und AutoCAD ab Version 2.6 deutsch oder englisch lauffähig.

Bauakademie der DDR, IPS/Abt. COMPAL, Plauener Straße 163–165, Berlin, 1092; Tel. 37 83 34 94
Piálek

LADAS

Das Dateibearbeitungsprogramm LADAS für KC 85/3 und /4 ist für die Bereiche Material- und Lagerwirtschaft, Kader- und Bildung, Produktionsplanung und -lenkung, W+T, Literaturrecherche u. v. a. m. geeignet. Es ermöglicht einen schnellen Zugriff auf Daten und dabei Dateiarbeit auf einfachster Anwenderebene. Das vorliegende Maschinencodeprogramm gestattet die Verwaltung von Dateien mit beliebig vielen Feldern und beliebigen Feldlängen. Die Satzzahl darf 2000 und die Satzlänge 16300 nicht überschreiten. Sollte der Anwender damit nicht ausreichen, so können mehrere Dateien mit gleicher Struktur, aber verschiedenen Namen angelegt werden. Die Anzahl der Dateien ist beliebig.
Das Programm LADAS ist bei Zusendung einer Magnetbandkassette oder einer formatierten MicroDOS-Diskette und eines formlosen Auftrages von uns für 800,– Mark zu beziehen.

Zwischengenossenschaftliche Einrichtung Landbauprojektierung Zwickau, Abt. Rechentechnik, Reichenbacher Straße 158, PSF 802, Zwickau, 9502; Tel. 78 82
Uhlig

Repräsentation von Meßsignalen mit LEMESS

LEMESS – in Turbo-Pascal 5.0 geschrieben – ermöglicht die komplexe Auswertung und Dokumentation beliebiger Meß- und Simulationsergebnisse, erleichtert Routinearbeiten und soll die Eigenentwicklung derartiger Programme erübrigen. Es dient zur Bestimmung der charakteristischen Kennwerte im Zeitbereich (z. B. Effektivwerte, Mittelwerte und Leistungen) und um Frequenzbereich (Spektren) aus Zeitfunktionen sowie der interaktiven grafischen Darstellung und Manipulation von Kurven jeder Art und damit einer detaillierten visuellen Meßkurvenanalyse sowie der Erarbeitung einer qualitativ hochwertigen grafischen Dokumentation.
LEMESS bietet (zur Zeit) folgende Möglichkeiten: individuelle Aufteilung des verfügbaren Datenspeichers, Erzeugung einfacher Testsignale, arithmetische Signalmanipulationen, Signalanalyse, Einbindung von Nutzerfunktionen sowie in LEMESS-Grafik die Darstellung von bis zu 32 Kurven in bis zu 9 frei definierbaren Fenstern mit variablem Fenster- und Kurvendesign, mit automatischer Skalierung sowie Achsenbeschriftung, Kursormessungen, Kurvenmanipulationen und Beschriften der Grafik „von Hand".
Als Datenquellen sind Simulationsprogramme (z. B. LENE) oder digitale

Meßgeräte (z. B. über V.24) denkbar, außerdem kann jedes Programm (in einer beliebigen Programmiersprache) Ergebnisse in kompatibler Form (ASCII-Files) auf Diskette schreiben.

Technische Universität Karl-Marx-Stadt, Sektion Automatisierungstechnik, PSF 964, Karl-Marx-Stadt, 9010; Tel. 561 36 10/36 12 *Waldeck*

Baumanzeige und Verzeichnisauswahl

Für Turbo-Pascal 4.0/5.0 wurde die allgemein verwendbare Unit TOMBAUM entwickelt. Diese UNIT kann mühelos in jedes Anwenderprogramm eingebunden werden, sie ermöglicht: Das aktuelle Laufwerk wird nach allen Directory-Einträgen durchsucht und als Liste (Baum) im Heap abgelegt. In diesem Baum wird nach einem übergebenen Verzeichnisnamen gesucht. Ist dieses Verzeichnis vorhanden, wird der komplette Pfadname und die Adresse, auf der sich dieses Verzeichnis befindet, an das Anwenderprogramm zurückgegeben. Die Funktion BaumAnzeigen realisiert die Ausgabe der Verzeichnisse auf den Bildschirm und die Verzeichnisauswahl mittels Kursortasten. Ein im Anwenderprogramm festgelegter Verzeichnisname kann vom Nutzer über diese Auswahl angelegt und als Adresse an das Anwenderprogramm zurückgegeben werden.
Geliefert werden die Programmdateien und eine Anwenderbeschreibung auf nutzereigenen 5,25-Zoll-Disketten (720 KByte).

VEB Datenverarbeitungszentrum Neubrandenburg, Bereich Absatz und Beschaffung, Koll. Gilberg, Straße der Befreiung 12, PSF 20, Neubrandenburg, 2000; Tel. 58 73 33 *Claus*

Druckertreiber für IFSS-Schnittstelle

Der Rechner A 7150 besitzt ein IFSS-Interface als zweiten seriellen Kommunikationsanschluß. Leider bedienen die Treiberprogramme des DCP nicht das XON/XOFF-Protokoll, welches zum Betrieb eines Druckers über IFSS notwendig ist.
Es wird kostenlos ein Treiber angeboten, welcher dieses Protokoll unterstützt. Der Datenverkehr wird mittels BIOS-Interrupt 14H durchgeführt. Durch die Anbindung an das BIOS ist der Treiber hardwareunabhängig.

Ulrich Zagorni, Hegemeisterweg 25, Berlin, 1157; Tel. 5 50 95 31, App. 33

JCPASCAL

JCPASCAL ist ein Pascal-System für Kleincomputer. Es zeichnet sich vor allem durch eine starke Anlehnung an Turbo-Pascal und eine Real-Arithmetik mit 16stelliger Genauigkeit aus. Für die Rechner KC 87 (KC 85/1), KC 85/2,/3,/4 und Z 1013 sind Versionen vorhanden. Die Datentypen sind gegenüber Standard-Pascal um BYTE, STRING (für Zeichenketten) und POINTER (ein allgemeiner Zeigertyp) erweitert. Verzichtet wurde dagegen auf Mengen- und Aufzählungstypen.
Hauptmenü und bildschirmorientierter Editor wurden ebenfalls an Turbo-Pascal orientiert gestaltet. Abweichungen resultieren vor allem aus den Besonderheiten des Kassetteninterfaces. Das Programmsystem hat eine Länge von etwa 16 KBytes.

Technische Universität Karl-Marx-Stadt, WB Tribotechnik 11270, PSF 964, Karl-Marx-Stadt, 9010 *Rödenbeck*

REDABAS für KC 87

Zur Lösung von Aufgaben der Kleinrationalisierung sowie zur Aus- und Weiterbildung habe ich das Programm REDABAS für den KC 87 entwickelt. Es realisiert zirka 70 % des REDABAS-Befehlsvorrats. Es hat eine Länge von rund 16 KByte und setzt 2 RAM-Module voraus. 16 Datenbanken oder Befehlsdateien können gleichzeitig im Speicher gehalten werden. Eine Datenbank besteht aus maximal 16 Feldern mit einer maximalen Länge von je 40 Zeichen. Die Satzlänge ist auf 200 Zeichen beschränkt. Es ist möglich, 16 Speichervariablen anzulegen. Die numerische Genauigkeit beträgt 10 Stellen. Druckertreiber werden automatisch berücksichtigt.
Das Programm mit Dokumentation kann an Interessenten kostenlos abgegeben werden.

Ronald Lippold, Wiesenstraße 22, Crimmitschau, 9630

Komfortabler Editor für KC87

Der Editor **QWORD** arbeitet screenorientiert. Er bietet eine Vielzahl von Möglichkeiten, die den Editoren von PCs oder BCs eigen sind. Sie können Texte einfügen und überschreiben sowie Blockkopier- und Blocktransportoperationen ausführen lassen. Texte werden automatisch rechtsbündig angeordnet und können gedruckt werden. Das Programm ist etwa 10 KByte lang und kann von Kassette geladen werden. Der Umfang bearbeitbarer Textdateien ist von der Speicherkonfiguration des KCs abhängig.

Technische Universität Dresden, Sektion Informationstechnik, Bereich Kommunikations- und Computertechnik, Mommsenstraße 13, Dresden, 8027; Tel. 4 63 51 21 *Herzmann*

Programme für Arbeit im DCP

Zur Verbesserung der Arbeit im Betriebssystem DCP entstanden in unserem Betrieb folgende Programme:
DHDATIM – menügesteuertes Setzen und Prüfen der Systemzeit.
DHF77I – Einfügen von INCLUDE-Texten in Fortran-Quellen: An die Stelle einer INCLUDE:...-Anweisung wird die INCLUDE-Datei selbst gesetzt. Damit wird dem Compiler die INCLUDE-Funktion abgenommen.
DHFORMAT – menügesteuertes Formatieren.
DHSETPRN – menügesteuertes Einstellen von Druckern: Das Programm ist in der Lage, einen Drucker der Typen DMP3160, FX1000, K 6313/6314, LX800 oder SL-80IP zu konfigurieren. Es ist möglich, weitere Druckertypen einzubinden.

VEB Datenverarbeitungszentrum Halle, Abt. FP, Block 081, Halle-Neustadt, 4090 *Dierich*

Ergänzungen zur DCP-Systemsoftware

Zur effektiven Arbeit unter dem Betriebssystem DCP bieten wir folgende zusätzliche Software an:
DSERV – ermöglicht problemloses Editieren von Diskettenfiles (Programmversion auch für SCP und SCP 1700).
DATETIME – ermöglicht unkompliziertes Setzen von Uhrzeit und Datum.
MAZEIT – realisiert eine rationelle Maschinenzeiterfassung, Datei ist mit REDABAS auswertbar.
RAMDISK – ermöglicht das flexible Einrichten eines virtuellen Laufwerkes.
FORMTXT – bietet folgende Möglichkeiten: Herausfiltern nichtdruckbarer Zeichen aus Texten, Umwandeln von Umlauten in Pseudoumlaute (ae, ...) und umgekehrt, Formatieren von Texten.
FRAGE – ermöglicht in Batch-Dateien das Formulieren von Abfragen und das Auswerten der darauffolgenden Bedienereingabe.
NCQUIT – ermöglicht das Verlassen des Norton-Commanders ohne Tastenbetätigung.

Institut für Forstwissenschaften Eberswalde, Organisations- und Rechenzentrum Potsdam, Koll. Neumann, Virchowstraße 39/41, Potsdam, 1591; Tel. 7 69 41 *Dr. Barciok*

Programmpaket „Feste Brennstoffe"

Das Projekt „Feste Brennstoffe" dient der rationellen Verwaltung aller Daten zu dieser Problematik und ist für die Bearbeitung eines Kohleplatzes ausgelegt.
Folgende gerätetechnische Voraussetzungen müssen erfüllt sein: PC 1715 unter SCP mit großem Bildschirm und 2 MFS 1.6, Drucker K 6313/SD 1152/LX 86.
Das Programmpaket existiert auch für 16-Bit-Technik (A 7150, EC 1834 oder analoges Gerät unter DCP und entsprechende Drucktechnik). Das Vorhandensein einer Festplatte wird vorausgesetzt.
Beide Projekte sind in Turbo-Pascal (Version 2.0 bzw. 4.0) geschrieben.

VdgB-Bezirksorganisation Karl-Marx-Stadt bei der BHG Auerbach, Bereich Rechentechnik, Bahnhofstr. 18, Auerbach, 9700; Tel. 30 33; Knauer *Schwabe*

Rechercheprogramm LIT

LIT ist ein sehr flexibel nutzbares Recherecheprogramm. Sieben Felder, maximal 99 Deskriptoren sowie ein weiteres Spezialfeld können optimal genutzt werden. Bezeichnungen, Längen und Positionierung sind mit Einschränkungen vom Nutzer frei wählbar. In einer so festgelegten Datenbank wird das Erfassen, Ändern und Löschen von Datensätzen unterstützt. Das Suchen von Datensätzen kann in allen Feldern nach Suchbegriff und – sofern beim Datenbankaufbau mit vorgesehen – nach maximal vier Deskriptoren erfolgen. In zwei Feldern ist auch eine schnelle indexsequentielle Suche möglich.
Das Programm kann auf XT-kompatiblen Computern mit Farb- oder Monochrommonitor genutzt werden. Voraussetzung sind mindestens 1 Diskettenlaufwerk sowie als Betriebssystem MS-DOS ab Version 3.1.

Institut für grafische Technik, Inselstraße 20, Leipzig, 7010; Tel. 6 86 32 55 *Dr. Fischer/Krüger*

GRUNDM

GRUNDM wurde mit PASCAL-T erarbeitet und liegt in Versionen für die Betriebssysteme SCP, SCP 1700 und DCP vor.
Das Projekt garantiert eine bedienerfreundliche Eingabe, Korrektur und Anzeige aller Datensätze. Die Robustheit wird u. a. durch eindeutige Modifizierungsstandsführung, Schutzcodeabfragen und Bereichsprüfungen gewährleistet.
Das Projekt wird gepflegt. Bei Bedarf kann eine betriebsspezifische Erweiterung vereinbart werden. Das Zusatzprogramm **GRZ** zur Zusammenfassung der Betriebsdateien kann ebenfalls genutzt werden. Eine umfangreichere Beschreibung wird Ihnen auf Anforderung kostenlos auf Ihrer 5,25"-Diskette (möglichst 16×256×80 initialisiert) übergeben.

Bezirkskombinat Fleischwirtschaft Neubrandenburg, Abt. MRT/EDV, Koll. Redlin, Steepenweg 18, PSF 17, Neubrandenburg, 2000; Tel. 5617 *Ehnert*

Stücklistenprogramm mit Stammdatenpflege

Wir bieten ein Softwarepaket zur Erarbeitung von Stücklisten für Elektro und HLS mit zyklischer Stammdatenpflege an. Bevorzugte Nutzergruppen sind Anwender des Softwarepaketes SLOBJEKT (Urheber ZBO Sömmerda) sowie Anwender, die bisher Stücklisten auf der Grundlage der TGA-Kataloge erzeugten. Einfache Handhabung des Programms, eine ausgefeilte Schlüsseltechnik und jederzeit aktuelle Stammdaten (gemäß TGA-Katalog) durch einen organisierten Änderungsdienst zeichnen diese Softwarelösung aus. Die Programmversion ist lauffähig unter SCP und bei Bedarf unter MS-DOS.

Zwischengenossenschaftliche Bauorganisation Landbau Gotha, Abt. Projektierung, Gartenstraße 52, Gotha, 5800; Tel. 5 38 08, 5 38 13 *Both*

Macroassembler für KCs

Der in MP 6/1989, Seite 187, angebotene Macroassembler läuft zusätzlich zu den dort angegebenen Computertypen auch auf dem KC 85/4. Inzwischen wurde auch eine CP/M-Version dieses Assemblers fertiggestellt.

Karl-Marx-Universität Leipzig, Institut für Biophysik, Liebigstraße 27, Leipzig, 7010 *Rödenbeck*

Wir suchen ...

... eine Hard- und Softwarelösung zum Anschluß eines Floppydisk-Laufwerkes 1.2 an einen Atari 800 XL.

Jugendklubhaus „Rosa Luxemburg", Zwickauer Straße 23, Altenburg, 7400; Tel. 31 53 *Richter*

Neues Echtzeit-Betriebssystem von Intel

iRMX III, das neue 32-Bit-Betriebssystem von Intel, soll vollständig kompatibel zu Intels jetzigen RMX-Betriebssystem-Versionen sein und den Systemintegratoren erlauben, die neue Generation der 32-Bit-Prozessoren und die dazugehörige Hardware voll auszunutzen. Um den Übergang zu Anwendungen mit voller 32-Bit-Leistungsfähigkeit zu unterstützen, wurden im Betriebssystem die Voraussetzungen dafür geschaffen, bestehende Software ohne Änderungen zusammen mit neu entwickelter 32-Bit-Software laufen zu lassen.
Das neue Betriebssystem ermöglicht auch den Zugriff auf die kürzlich von Intel vorgestellte Familie von 32-Bit-Peripheriebausteinen. So sind die neuen 32-Bit-E/A-Controller bei höherer Kapazität schneller als ihre Vorgängertypen, jedoch vollständig kompatibel dazu. Einschränkungen durch Segmentgrößen gehören jetzt der Vergangenheit an. Mit iRMX III lassen sich Segmentgrößen von 1 Byte bis zu 4 GByte konfigurieren. Mehr als 8000 dieser Segmente können von dem neuen Betriebssystem verwaltet werden, wobei die Anwender vom Intra-Segment-Schutz des 386 profitieren.
iRMX III unterstützt auch die volle 32-Bit-Arithmetik und erlaubt den Einsatz des Arithmetik-Koprozessors 80 387. Dabei sind Leistungssteigerungen um den Faktor 2 bis 5 zu erwarten – speziell bei der Ausführung komplexer Funktionen, die wie SIN, COS und LOG früher von dem Arithmetik-Koprozessor 80 287 in Software emuliert wurden. Diese Verbesserung ist besonders für Datenerfassungsanwendungen interessant, die oft komplexe Skalierfunktionen mit den hereinkommenden Daten ausführen.
Das Betriebssystem iRMX III wird für alle drei Intel-Architekturen verfügbar sein – mit voller Netzwerkunterstützung durch RMXNET, Grafikunterstützung mit RGI auf dem iSBX 279 und unterstützt durch das Spektrum bereits verfügbarer 32-Bit-iRMX-Entwicklungswerkzeuge. *MP*

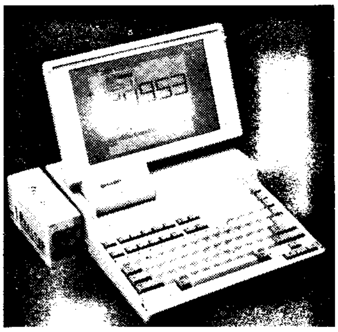
Siemens und IBM entwickeln gemeinsam 64-MBit-Chip

Die Siemens AG und IBM haben vereinbart, bei der Entwicklung des 64-MBit-DRAMs (Dynamic Random Access Memory) zusammenzuarbeiten. Die Vereinbarung beinhaltet eine mögliche Erweiterung auch auf nachfolgende Chip-Generationen. Beide Partner tragen die Entwicklungskosten je zur Hälfte.
Die Entwicklungsarbeiten für den 64-MBit-DRAM werden bei IBM und Siemens unverzüglich aufgenommen und in dem neu errichteten Advanced Semiconductor Technology Center in East Fishkill konzentriert. Sie bauen auf den Forschungsaktivitäten von Siemens in München und IBM in Essex Junction, USA, auf.
Ziel der Vereinbarung ist es, bis Mitte der 90er Jahre weltmarktfähige 64-MBit-DRAMs produzieren zu können. Auch wenn beide Partner beschlossen haben, die Kosten des Projekts vertraulich zu behandeln, gehen doch Fachleute davon aus, daß eine solche Entwicklung etwa 450 Mio. US-$ kosten könnte. *MP*

Framework III – Anwendungsprogrammierung

Für die in Framework integrierte Programmiersprache FRED bietet Ashton-Tate jetzt eine Sammlung von Programmen an, mit der Entwickler professioneller Software Framework III-Anwendungen einem großen Kreis von Nutzern zugänglich machen können. Die Sammlung beinhaltet unter anderem Anpassungsprogramme für Drucker und Plotter, Bildschirmtreiber und Programmteile zur erweiterten Nutzung von Framework III mit detaillierter Dokumentation. Die Einbindung von C- und Assemblerroutinen in FRED-Programme ist ebenfalls möglich. Das mitgelieferte Runtime-Modul gestattet es, Anwendungen unabhängig von Framework einzusetzen. Die Programmsammlung und das Runtime-Modul werden von Ashton-Tate für 499.– DM angeboten. *MP*

Vom PC zum digitalen Meßgerät

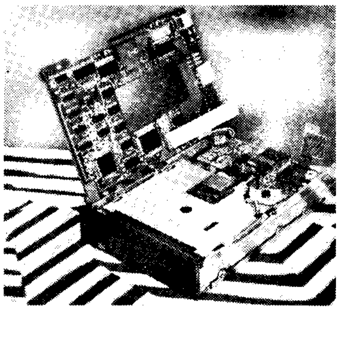


Eine vom Hersteller ttw GmbH vorgestellte Neuentwicklung soll völlig neue Möglichkeiten im Bereich der digitalen Meßtechnik eröffnen. Der Grundgedanke des Practop-Systems: Messen und die digitale Aufbereitung der Daten sind getrennt. Die Meßmodule – zum Beispiel Multimeter, Logikanalyzer, Oszilloskop – erfassen lediglich Daten, das Anzeigen und Auswerten übernimmt ein PC. Im Gegensatz zu herkömmlichen Meßkarten werden die Module jedoch nicht in den Rechner eingebaut, sondern einfach von außen an die serielle Schnittstelle RS232 des PCs angesteckt. Dabei ist jedes Modul eine autarke Meßeinheit, das Daten selbständig aufnimmt und intern abspeichern kann.
Das neue Konzept „Meßmodul + PC" vereinfacht insbesondere Wartungs-, Reparatur- und Servicetechnikern die Arbeit beim Kunden. Ein Laptop, die Meß-Software und eine Tasche mit den Practop-Modulen ersetzen mehrere herkömmliche digitale Meßgeräte.
Für die Arbeit mit Practop braucht man kein spezielles Computerwissen. Der Rechner erkennt automatisch, welches Modul angeschlossen ist. Ein übersichtliches Menü in Windowtechnik erleichtert die Benutzerführung.
Das System ist so ausgelegt, daß der Anwender auch eigene Software-Anpassungen und Erweiterungen entwickeln kann. Zudem bietet ttw seinen Kunden Zusammenarbeit bei der Anpassung an spezielle Kundenanforderungen und Branchenlösungen an, und zwar sowohl auf Software- als auch auf Hardware-Ebene *MP*

Pagemaker 4.0

Die Version 4.0 des Pagemaker der Firma Adobe soll seit April 1990 verfügbar sein und sich durch völlig neue Funktionen auszeichnen. Insbesondere für die Textverarbeitung sind jetzt Funktionen zum Suchen und Ersetzen, zur Trenn- und Rechtschreibkontrolle nach einem deutschen Lexikon mit 150 000 Wörtern und zur automatischen Generierung von Inhalts- und Indexverzeichnissen enthalten. Zur typografischen Gestaltung werden ebenfalls neue Funktionen angeboten: Freie Wahl der Zeichen-, Wort- und Absatzabstände, Einbinden mitfließender Grafiken, Buchfunktionen mit dateiübergreifender Seitennumerierung, Textrotation in 90°-Schritten, Unterschneidung in Schritten von 1/100 Geviert, einstellbare „Schusterjungen"- und „Hurenkind"-Funktion und anderes. Das Spezialprogramm Aldus Preprint dient der Vierfarbseparation und ermöglicht unter anderem die Farb- und Bildkorrektur sowie die Definition von Rasterwinkeln und Rasterweiten für jede Farbe auch für alle fremden Dateien, sofern sie die Adobe-Postscript-Farbkonvention erfüllen. *MP*

20-MByte-Floppy statt Festplatte



Die britische Firma Citizen hat jetzt das nach eigenen Angaben erste serienmäßige 20-MByte-Diskettenlaufwerk der Welt vorgestellt. Es soll vor allem in Laptops eine Alternative zu den Festplattenlaufwerken sein. Das im Bild gezeigte neue 3,5-Zoll-Laufwerk hat eine Bauhöhe von 1 Zoll und verfügt über eine SCSI- und eine AT-Bus-Schnittstelle. Die mittlere Zugriffszeit beträgt 50 ms. Mittels Schwingspulenmotor und optischer Servospuren auf der Diskette wird die exakte Positionierung des Schreib-/Lesekopfes erreicht. Zur Zeit können mit dem Laufwerk auch 1- oder 2-MByte-Floppies gelesen werden; die Schreib-/Lesefunktion soll in Kürze folgen.
Da von mehreren PC-Herstellern in diesem Jahr bereits PCs mit 4-MByte-Diskettenlaufwerken erwartet werden, bietet Citizen jetzt ebenfalls derartige 3,5-Zoll-Laufwerke in den Bauhöhen 3/4 Zoll und 1 Zoll an. *MP*

Löschbare optische Platte mit 1 GByte

Die Firma Storage Dimensions hat eine Serie von optischen Plattensystemen angekündigt, deren wechselbare Datenträger (Kassetten) Kapazitäten bis zu 1 GByte besitzen. Die durchschnittliche Zugriffszeit beträgt 35 ms, die Übertragungsgeschwindigkeit 10 MBit/s. Die Systeme lassen sich an MS-DOS-Rechner (ab 80 286-Prozessor) und Macintoshs anschließen. Um entsprechend der ISO-Norm Kompatibilität zu anderen Laufwerkherstellern zu gewährleisten, gibt es auch eine Version mit 650 MByte. *MP*

Microsoft LAN-Manager/X für OEM-Kunden

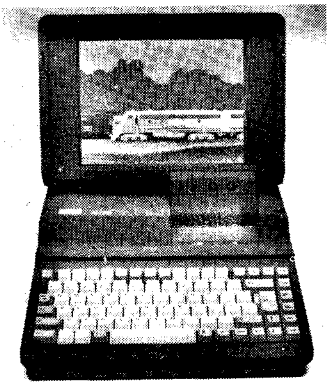
Nachdem Microsoft den LAN-Manager/X an seine OEM-Kunden ausgeliefert hat, kann damit gerechnet werden, daß die ersten Produkte in Kürze zur Verfügung stehen. Die portable Version des OS/2 LAN-Managers wurde gemeinsam mit Hewlett Packard entwickelt und erlaubt Anwendern von PCs, Mini- und Großrechnern die gemeinsame Nutzung von Dateien, Programmen und Peripheriegeräten. Zahlreiche Firmen wollen den LAN-Manager/X in Zukunft unterstützen, so neben Hewlett Packard beispielsweise auch Bull, AT&T, Santa Cruz Operation, Tandem und Sybase. Der LAN-Manager/X deckt einen breiten Hardwarebereich ab und erlaubt es, die Vorteile der Client-Server-Architektur von Microsoft auf heterogene Umgebungen zu übertragen. Besonders vorteilhaft ist dabei die offene Systemarchitektur, die Kompatibilität und die Anpassung an den Industriestandard. Mehrere Protokolle wie beispielsweise TCP/IP, ISO, NetBEUI und XNS werden vom LAN-Manager/X unterstützt. Des weiteren wird die Kommunikation verteilter Applikationen über das Netz ermöglicht. Neben einer vollen Hotline-Unterstützung stellt Microsoft Beispiel-Software unter anderem für das Unix System V/386, für MS-/PC-DOS und OS/2 zur Verfügung *MP*

Neue grafische Benutzeroberfläche

Mit dem HP Visual User Environment (HP VUE) will Hewlett-Packard den Umgang mit dem Betriebssystem UNIX erleichtern. Die grafische Benutzeroberfläche bietet einen Vorrat leicht verständlicher Symbole (Icons) und eine Reihe von Dienstprogrammen einschließlich File-Manager, Hilfe und Mailer sowie Text- und Bitmap-Editoren.
HP VUE basiert auf dem X Window System in der Version 11 sowie OSF/Motif und damit auf offenen Industriestandards. Es ist ein erster Schritt zur

Implementierung der Software-Umgebung HP NewWave in der UNIX-Welt. Das Verhalten entspricht dem des HP Presentation Managers. *MP*

Amstrad-Laptops



Den Wettbewerb auf dem Laptop-Markt kräftig anregen will Amstrad mit einem 286er und einem 386SX-Laptop zum Preis von 5500 bzw. 6900 DM. Beiden Geräten sind 16 MHz Taktfrequenz, eine Festplatte mit 20 oder 40 MByte, ein 1,44-MByte-Diskettenlaufwerk und das Backlit-Supertwist-LCD mit 640 × 480 Punkten gemeinsam. Der 1-MByte-Hauptspeicher ist erweiterbar auf 4 MByte. Der Grafikadapter ist nicht nur zur VGA, sondern auch zu EGA, CGA, MDA und Hercules kompatibel. Die 256-Farben-Palette wird in 32 Graustufen umgesetzt. Der ALT 286 und der ALT 386SX sind Eigenentwicklungen, die bei Vertragspartnern in Japan komplett gefertigt werden. Amstrad will mit dem Laptop nicht nur Einsteiger, sondern auch professionelle Anwender erreichen. *MP*

EDOS – 736 KByte RAM unter MS-DOS

Oberhalb der 640-KByte-Grenze von MS-DOS stellt das von Ashton-Tate vertriebene DOS-Tool EDOS weitere 96 KByte RAM zur Verfügung. Auf allen PCs (XT/AT), PS/2 und dazu kompatiblen Systemen mit 640 KByte RAM und mindestens 256 KByte Videospeicher (EGA oder VGA) soll EDOS problemlos einsetzbar sein. Eine der üblichen Speichererweiterungen wie Extended Memory oder Expanded Memory sind hierfür nicht erforderlich. Ashton-Tate bietet EDOS für 49.– DM an. *MP*

Seagate Wren Runner 2

Mit einem neu entwickelten Spindelmotor – mit 4800 Umdrehungen pro Sekunde – hat der Festplattenhersteller Seagate die Leistungsfähigkeit seiner Wren-Runner-Serie gesteigert. Das Laufwerk Wren Runner-2 ST4767N ist eine 5,25-Zoll-SCSI-Festplatte mit 767 MByte unformatierter Kapazität. Seagate hebt vor allem die geringe durchschnittliche Zugriffszeit von 11,9 Millisekunden und die Datentransferrate von 24 MBit/s hervor. Das Laufwerk mit voller Bauhöhe ist insbesondere für den Einsatz in Workstations und mittleren DV-Systemen gedacht. *MP*

Hitachi auf dem Weg zum 1-GBit-DRAM

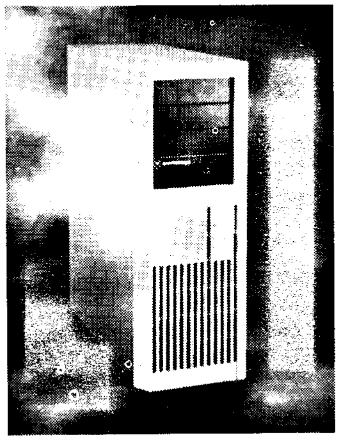
Herkömmliche Transistoren werden durch das Bearbeiten einer Siliziumscheibe von oben in der sogenannten Planar-Technik hergestellt. Die japanische Firma Hitachi hat nun – abweichend von dieser Technologie – einen würfelförmigen Transistor entwickelt, mit dem man eine 250fach höhere Integrationsdichte als beim 4-MBit-Speicher erreichen will. Auch die Geschwindigkeit dieses *Delta-Transistors* soll 2- bis 3mal höher sein. Laut Hitachi soll der Delta-Transistor den Weg zum 1-GBit-DRAM ebnen. *MP*

Intel erweitert 16-Bit-Embedded-Controller-Familie

Intels Unternehmensbereich Microcomputer-ASIC in Chandler stellte im März dieses Jahres sieben neue Mitglieder seiner 16-Bit-Mikrocontroller-Familie MCS-96 zum Einsatz in eingebetteten Echtzeitsystemen vor. Die Mikrocontroller bieten sowohl eine höhere Leistung als auch eine größere Integrationsdichte als die bisher bekannten 80C196KB-Mikrocontroller, die seinerzeit (1988) die ersten MCS-96-Produkte in CMOS-Technologie waren (vergleiche MP 5/90, S. 130).
Zwei neue Versionen, der 80C194 und der 80C198, erlauben es dem Anwender, bei einer internen Verarbeitung von 16 Bit mit einem externen 8-Bit-Bus die vorhandene 8-Bit-Peripherie anzusprechen. Der entscheidende Unterschied zwischen beiden ist ein Analog-Digital-Wandler, der beim 80C198 integriert ist, beim 80C194 jedoch nicht.
Weitere neue Hochleistungsmikrocontroller dieser Familie sind folgende:
– der 80C196KC und der 80C196KR mit einem auf dem Chip integrierten 16-KByte-EPROM/-ROM
– der 80C196JR, eine pinreduzierte und „abgemagerte" Version des 80C196KR
– der 80C196KQ mit 12 KByte ROM oder EPROM auf dem Chip
– der 80C196JQ, eine pinreduzierte und „abgemagerte" Version des 80C196KQ.
Die Controller basieren auf Intels High-Density-1μ-CHMOS-Prozeß.
Intels Unternehmensbereich Automotive bietet alle neuen Mikrocontroller-Versionen im vollständigen Automobiltemperaturbereich von −40 bis +125 Grad Celsius an. *MP*

Motorola mit RISC-PC für Einsteiger

Bei Motorola werden nicht nur Bauelemente, sondern auch äußerst leistungsfähige Computer entwickelt. Beispielsweise kommt die Serie Delta 8000 RISC von der Firma Motorola Computer Systems. Kernstück dieser Rechner ist der RISC-Prozessor 88000 von Motorola (vgl. MP 11/88, S. 351 und MP 11/89, S. 348). Die Serie Delta 8000 RISC wurde kürzlich um das Modell 8408 erweitert, das vorrangig auf Einsteiger zielt. Dem potentiellen „Einsteiger" werden beachtliche 17 MIPS Leistung bei einer Taktfrequenz von 20 MHz, 8 MByte Hauptspeicher und 300 oder 600



MByte Plattenspeicherkapazität geboten. Der Rechner, der über 6 Erweiterungssteckplätze verfügt und an den 64 Nutzer anschließbar sein sollen, wird für rund 27 000 bzw. 28 000 Dollar angeboten. *MP*

Festplattenlaufwerke mit Gigabyte-Kapazität

Von Seagate wurde kürzlich der als Festplattenspezialist bekannte Hersteller Imprimis übernommen. Infolgedessen bietet Seagate nunmehr Festplattenlaufwerke mit außerordentlich hoher Kapazität an. Die Serie Elite beispielsweise bietet bei 12 ms durchschnittlicher Zugriffszeit 1,5 GByte Kapazität und wird wahlweise mit den Schnittstellen SCSI, SMD oder IPI ausgestattet. Sie ist damit insbesondere für Workstations geeignet. *MP*

Winchesterlaufwerk mit 2 GByte

Einen außergewöhnlich schnellen 8-Zoll-Winchester-Speicher mit der gleichzeitig großen Speicherkapazität von 2,027 GByte stellte die japanische Firma Fujitsu vor. Die Zugriffszeit liegt bei 12 ms, die Übertragungsrate bei 3 MByte pro Sekunde. Gegenüber 5,25-Zoll-Platten sollen der Raum- und der Energiebedarf geringer sein. *Wi*

Keine Gefährdung durch Bildschirmterminals

Die verbreitete Anwendung von Bildschirmterminals hat ständig zu Diskussionen über gesundheitliche Gefährdungen geführt. Haltungsschäden, Magenleiden und Auswirkungen von Röntgenstrahlen waren bei Geräten der älteren Generation der Hauptangriffspunkt. Neuere Geräte sind ergonomisch so gestaltet, daß diese Gefährdungen nicht auftreten.
Untersuchungsergebnisse schwedischer Wissenschaftler über Gesundheitsgefährdungen durch niederfrequente Strahlen und elektromagnetische Felder haben die Diskussionen wieder belebt. Bei Mäusen wurden Wachstumsstörungen und Mißbildungen beobachtet. Die These, daß bei schwangeren Frauen negative Auswirkungen auf das ungeborene Kind auftreten, konnte praktisch und statistisch widerlegt werden. Durch Haushaltgeräte ist die schwangere Frau in weitaus größerem Umfang niederfrequenten elektromagnetischen Feldern ausgesetzt, so daß die Strahlung von Terminals nicht allein als Quelle der gesundheitlichen Risiken auftritt. Die niederfrequenten, langwelligen Felder (bis 300 000 Hz) schwanken stark in ihrer Intensität. Als gesundheitliche Risiken werden Schlafstörungen, Angstzustände bei Säuglingen, Gereiztheit, Herzrhythmusschwankungen, Förderung des Krebswachstums und Nieren- und Darmkoliken genannt.
Als Maßnahme wird eine Abschirmung durch eine geerdete Kupfer- oder Aluminiumfolie um das Gerät vorgeschlagen. Eine Folie vergrößert jedoch die Gefahr der Überhitzung des Terminals.
LCD-Bildschirme vermeiden aufgrund ihres physikalischen Wirkungsprinzips niederfrequente und elektromagnetische Strahlungen. Sie besitzen jedoch Nachteile durch das lange Nachleuchten der Zeichen.
Bildschirme werden in den einzelnen Ländern einer Prüfung unterzogen. Bewährt hat sich die skandinavische Meßmethode. Bei ihr wird das elektromagnetische Feld in Abständen von 30 und 70 cm von der Bildmitte gemessen. Im Abstand von 50 cm werden kugelförmig um das Gerät der höchste und niedrigste Meßwert festgestellt. Der Maximalwert sollte 20 mT/s nicht überschreiten.
Die modernen Bildschirmterminals liegen in ihren Werten deutlich unter den Grenzwerten. Der maximale Wert des elektrostatischen Feldes wurde in skandinavischen Normen bei 50 kV/m festgestellt. Moderne monochrome Terminals liegen bei 12 kV/m.
Eine weitere Gesundheitsgefährdung kann durch Flammenhemmer auftreten, die in Elektrogeräten Anwendung finden, um Kunststoffbrände zu vermeiden. Diese Flammenhemmer sind stark giftig (Decabromdiphenylether) oder mit Giften verunreinigt (polybromiertes Dibenzufuran). Bei wechselnder thermischer Beanspruchung diffundieren die Gifte. Dies kann sich über einen längeren Zeitraum hinziehen.
Als schleichende Vergiftungserscheinungen treten häufig Infektionen oder Nervenleiden auf (mit Symptomen wie verstopfte Tränenkanäle, trokkene Augen, Nachtblindheit, Entzündungen von Leber und Bauchspeicheldrüse, Arteriosklerose, Verengung der Herzkranzgefäße).
Seit 1986 sind diese Erscheinungen bekannt, und die Kunststoffhersteller nehmen seither von diesen Chemikalien Abstand. Da bei älteren Geräten die Giftstoffe bereits entwichen sind, dürfte nach Ansicht der Verfasserin bei heute in Betrieb befindlichen Geräten keine Gefahr mehr bestehen. *Wi*

Hinweis

Sicher haben auch Sie bemerkt, daß die in MP 5/90, Seite 158, abgebildete neue Siemens-Workstation WX 200 kein Laptop ist. Durch ein Versehen sind lediglich die Fotos der WX 200 und des Sharp-Laptops PC-6220 vertauscht worden.

Datenübertragung leicht gemacht?

International längst üblich: Mailboxen statt Eintippen von Hexdumps, Online-Datenbanken statt Bücherwälzen, elektronische Postwurfsendungen statt blendfreier Kontaktarmut. Ein Modem muß her. MP machte sich zu den rechtlichen und technischen Voraussetzungen im Institut für Post- und Fernmeldewesens der Deutschen Post sachkundig.

Problematisch bei der Nutzung unseres Fernsprechnetzes für die Datenübertragung ist die schlechte Leitungsqualität (Fehlerraten im Bereich von bis zu 10^{-3}). Außerdem ist die Dämpfung durch die im Selbstwählfernverkehr im öffentlichen Fernsprechnetz eingesetzte Gabel 64 zu beachten. So darf sich innerhalb einer Verbindung nur ein TF-Primärgruppenabschnitt (Umsetzer 2-Draht/4-Draht) befinden. Aus den genannten Gründen ist eine sichere Übertragung nur bis 300 Bit pro Sekunde möglich; bei Modems mit dem fehlerkorrigierenden Verfahren Microcom Networking Protocol (MNP) kann mit 2400 Bit/s gearbeitet werden. Das Telexnetz wird mit 50 Bit/s betrieben. Für höhere Übertragungsraten steht das handvermittelte Datennetz mit Leitungen für 2400 Bit/s zur Verfügung. Besser ist die Lage bei den *überlassenen Übertragungswegen* (umgangssprachlich Postmietleitungen). Sie können mit 9600 Bit/s betrieben werden. In geringem Umfang sind auch größere Geschwindigkeiten (48 kBit/s) möglich. Im Aufbau befindet sich ein paketvermitteltes Datennetz (nach X.25-Spezifikation), das in der ersten Ausbaustufe 4000 Teilnehmer umfassen soll. Vorerst wird es sich um ein Datenstandleitungsnetz für 1000 Teilnehmer handeln, weil entsprechende Vermittlungstechnik gegenwärtig noch nicht verfügbar ist.
Die Leistungen der Deutschen Post bestehen in der Bereitstellung von Datenanschlußleitungen in vermittelten Netzen und in der Überlassung von Übertragungswegen. Die vom Teilnehmer zu beschaffende, einzurichtende und zu unterhaltende Datenübertragungseinrichtung muß von der Deutschen Post für den Anschluß an das Fernmeldenetz zugelassen werden. „Betriebe, Einrichtungen und Massenorganisationen" können einen Antrag auf Zulassung bei der Deutschen Post stellen. Für die in Tafel 1 aufgeführten Modems liegt eine typengebundene Zulassung der Deutschen Post vor, die vom jeweiligen Hersteller beantragt wurde. Aus naheliegenden Gründen wird das Angebot an diesen Geräten stark steigen.
Für alle anderen Typen muß eine einzelerzeugnisgebundene Zulassung beantragt werden, das heißt, das betreffende Gerät wird für einen bestimmten Anwendungsfall überprüft und erhält für diesen eine Zulassung. Dazu ist ein formloser Antrag an das Institut für Post- und Fernmeldewesen der DDR, Abt. FD3, Oranienburger Straße 70, Berlin, 1040, zu stellen, der folgende Angaben enthalten muß:

- Typbezeichnung und Herstellerfirma des Gerätes
- Anschaltung: öffentliches Fernsprechnetz, handvermitteltes Datennetz oder überlassener Übertragungsweg
- Verkehrsbeziehung: Adressen der kommunizierenden Partner
- Übertragungsgeschwindigkeit
- Betriebsweise: synchron oder asynchron
- Betriebsverfahren: simplex, halbduplex oder duplex
- Anzahl der anzuschließenden Geräte.

Für die Zulassungsuntersuchungen sind der Deutschen Post zwei Geräte und eine komplette Dokumentation (Funktionsbeschreibung, Betriebs-, Instandhaltungsvorschrift und Stromlaufpläne) in deutscher oder englischer Sprache zur Verfügung zu stellen. Die Dokumentation verbleibt bei der Deutschen Post. Die Untersuchungen für einzelerzeugnisgebundene Zulassungen dauern 14 Tage, gerechnet vom mit der Deutschen Post abgestimmten Übergabetermin.
Für das Zulassungsverfahren werden aufwandsabhängig Gebühren erhoben und dem Antragsteller in Rechnung gestellt.
Für das Betreiben von Modems bzw. Basisbandübertragungseinrichtungen am Fernmeldenetz der Deutschen Post ist neben der Zulassung der Geräte (Zertifikat über die einzelerzeugnisgebundene bzw. typgebundene Zulassung) die Anschaltung an das Fernmeldenetz zu beantragen. Dafür gelten die in Tafel 2 aufgeführten Zuständigkeiten.
Unabhängig von der Art der Zulassung besteht die Pflicht zur Vorführung der Geräte bei der Inbetriebnahme, das heißt, dem Mitarbeiter der Deutschen Post ist der Sendepegel vorzumessen und die richtige Arbeitsweise der Modems anhand eines Übertragungsversuches nachzuweisen.
Akustikkoppler werden nur eingeschränkt genehmigt, nämlich für den nichtstationären Betrieb.
Für die Datenübertragung im öffentlichen Fernsprechnetz, im handvermittelten Datennetz und auf überlassenen Übertragungswegen bildet der leitungsmäßige Ausgang des Modems bzw. der Basisbandübertragungseinrichtung die Schnittstelle zum Fernmeldenetz der Deutschen Post, so daß beliebige Endgeräte (Computer) angeschlossen werden können (im Gegensatz zu Fernkopierern und anderen direkt angeschlossenen Endgeräten, die als solche genehmigt werden müssen).

Wir danken Frau Dr. Renneberg vom Institut für Post- und Fernmeldewesen für die umfassende Auskunft und freundliche Unterstützung.

Tafel 1 Modems mit typengebundener Zulassung, Stand: 25. Mai 1990

Typ	Hersteller	Bit/s
AM 2400	Firma Orion (Ungarn)	2400/1200
AM 1201	Firma Orion (Ungarn)	1200/600
VM 2400	VEB RFT NEL	1200/600
TAM 1200	Fa. Telefongyar (Ungarn)	1200/600
AM 12TD	Firma Orion (Ungarn)	1200/600
AM 1200	Firma Orion	1200/600
DNÜ 9600	VEB Robotron-Elektronik	9600 *)
TAM 300	Fa. Telefongyar (Ungarn)	300
VT 60200	Firma Videoton	600/1200
VT 60005/60004	Firma Videoton	300
VT 60205/60204	Firma Videoton	600/1200
VT 60305/60304	Firma Videoton	1200/2400
VT 61505/61504	Firma Videoton	200 bis 9600 *)
MicroLink 2400 T2 mit MNP	Motorola/Codex (USA) und Elsa (BRD)	2400/1200/300
Codex 2345	Motorola/Codex (USA) und Elsa (BRD)	9600/4800/2400
Omnimode 96	Racal Milgo (GB)	9600/4800/2400
MPS 9601	Racal Milgo (GB)	9600/4800/2400
Fury 2400 TI u. PC mit MNP	Dr. Neuhaus GmbH (BRD)	2400/1200/600/300
Alpha 96 Plus	Racal Milgo (GB)	9600/4800/2400

*) Basisbandübertragungsrichtung

Tafel 2 Zuständigkeiten

Datenübertragung auf überlassenen Übertragungswegen	Antrag auf Anschaltung an
innerhalb des Bezirkes	Fernsprechanmeldestelle des Post- und Fernmeldeamtes (PFA)
zwischen mehreren Bezirken	ZPF, Abt. Fernmeldenetze, Sektor Bedarfskoordinierung Leipziger Str. 8 3014 Magdeburg
international	MPF, Abt. BVFe Mauerstraße 69–75 1066 Berlin

Für das handvermittelte Datennetz und das Telexnetz sind die Anträge an das zuständige Post- und Fernmeldeamt – in Berlin an das Fernsprechamt für Fernsprechanschlüsse und an das Fernamt für Telexanschlüsse – zu richten.

Tafel 3 Gesetzliche Bestimmungen

Gesetzblatt der DDR Sonderdruck 1268/1986 vom 18. April 1989
Anordnung über den Datenübertragungsdienst – Datenübertragungs-Anordnung – vom 28. Februar 1986
Anordnung über leitungsgebundene Fernmeldeanlagen für den nichtöffentlichen Fermeldeverkehr und für das Überlassen von Übertragungswegen vom 28. Februar 1986
Gesetzblatt der DDR, Teil I Nr. 31 vom 09. Dezember 1985
Gesetz über das Post- und Fernmeldewesen vom 29. 11. 1985
Durchführungsverordnung zum Gesetz über das Post- und Fernmeldewesen – Genehmigung zum Fernmeldeverkehr – vom 29. 11. 1985

vorgestellt

TEXgraf

Grafische Schaltungserfassung im ASIC-Entwurf

Möglichkeiten zur interaktiven grafischen Arbeit sind integraler Bestandteil moderner CAD-Systeme. Für die Schaltplanerfassung beim Entwurf von Gate-Array-Schaltkreisen im System U 5200 wurde bisher lediglich eine alphanumerische Eingabe (NBS 84) vorgesehen. Gegen eine grafische Eingabe wurden folgende Argumente vorgebracht:

– höherer Zeitaufwand für die reine Schaltplaneingabe
– wesentlich höhere Gerätekosten und damit verbunden eine meist unbefriedigende Verfügbarkeit.

Das erste Argument ist letztlich wenig überzeugend, da nur ein Teilschritt des Entwurfsprozesses betrachtet wird. Bei einer rechnergestützten Schaltplanerfassung sind jedoch Effektivitätssteigerungen im Gesamtprozeß der Schaltungsentwicklung möglich:

– In der kreativen Phase werden die benötigten Schaltelemente mit dem Grafiksystem auf einem „Blatt" angeordnet und ausgeplottet. Mit diesem „Gerüst" erfolgt dann die Erarbeitung der vollständigen Schaltung. Die Einsparung an Zeichenarbeit ist im allgemeinen beachtlich.
– Ohne zusätzlichen Aufwand entsteht eine projektbegleitende und stets aktuelle Dokumentation.
– Bei grafischer Schaltplaneingabe werden elementare Eingabefehler vermieden, die für die alphanumerische Eingabe typisch sind. Zusätzlich kann ein spezielles Grafiksystem sofort jede Eingabe nach Plausibilitätskriterien überprüfen.
– Der Eingabeaufwand kann in speziellen Schaltungseditoren auf ein effektives Mindestmaß reduziert werden. Selten genutzte Gestaltungsfreiheiten werden als Sonderfunktionen angeboten.
– Die manuelle Extraktion einer alphanumerischen Schaltungsbeschreibung (NBS) aus der entworfenen Schaltung entfällt.
– Neben der grafischen Schaltungserfassung bietet sich auch eine grafische Darstellung von Simulationsergebnissen an. Damit wird der traditionelle Arbeitsstil des Ingenieurs unterstützt.

Das zweite vorgebrachte Argument verliert seine Bedeutung mit der Bereitstellung preiswerter PCs mit geeigneten Eigenschaften.

Da universelle Schaltungseditoren diese vorteilhaften Effekte im allgemeinen nicht vollständig bewirken, wurde im VEB Textimaelektronik der Schaltungseditor *TEXgraf* für das Gate-Array-Entwurfssystem U 5300 des VEB ZMD entwickelt.

TEXgraf existiert in mehreren Ausführungen:

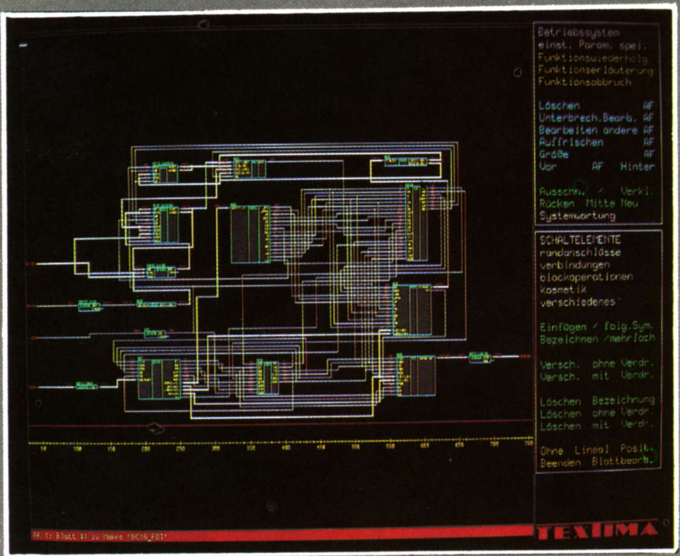
1. Der grafische Schaltungseditor ist als Funktionsmodul in das Entwurfssystem ARCHIMEDES für U 5300 integriert. TEXgraf läuft in diesem Fall auf dem 32-Bit-Rechner K 1840 unter SVP 1800 in Verbindung mit dem Grafikterminal K 8919.11 oder kompatibler Technik. Darüber hinaus werden Terminals mit 12-Bit-Koordinatenraum unterstützt. Die konkrete Gerätekonfiguration ist vom Betreiber in einer Gerätekonfigurationsdatei anzugeben (grafisches Eingabegerät, Verfügbarkeit eines parallelen Interfaces).
2. TEXgraf ist weiterhin als autonomes System auf dem K 1840 vorhanden und kann in dieser Version auch für den Entwurf von Standardzellen-ASICs (U 1500, U 1600) verwendet werden.
3. Um dem Schaltungsentwickler TEXgraf kostengünstiger als Werkzeug für den Arbeitsplatz bereitzustellen, wurde eine Version für PCs der AT-Klasse (oder höher) mit EGA- oder VGA-Grafiksteuerung unter MS-DOS geschaffen. Als Eingabegerät ist eine Maus mit Microsoft-Schnittstelle erforderlich. Im Vergleich zu einem K 1840/K 8919.11 im „durchschnittlichen" Mehrnutzerbetrieb ergeben sich bei einem PC-AT mit 10 MHz Taktfrequenz für die meisten Funktionen Geschwindigkeitsvorteile (Ausnahme: Auffrischen von Arbeitsflächen). Nachteilig ist die verringerte Bildschirmauflösung von etwa 50 Prozent, was aber durch die Ausschnittsfunktion kompensiert werden kann.

Der Schaltungseditor ist durch folgende Merkmale charakterisiert:

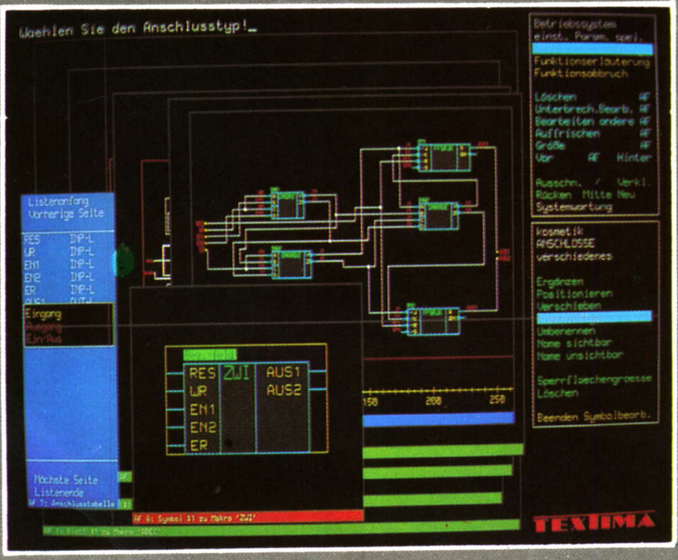
• *hierarchische Strukturierung*
In Anlehnung an bekannte Praktiken der Untergliederung komplexer Systeme nach dem Hierarchieprinzip können Teilschaltungen (Anwendermakros) definiert werden, die in einer hierarchisch höheren Schaltung als Symbol auftreten. Als Elementare sind die Hardwaremakros des ASIC-Systems U 5300 implementiert. Zusätzlich ist die Verteilung eines Schaltplanes auf mehrere Blätter möglich.

• *Datenabspeicherung/Schnittstellen*
In der ARCHIMEDES-Version werden alle Daten im Kundenspeicher des Entwurfssystems abgelegt. Die PC-Version von TEXgraf arbeitet mit Grafikdateien, in denen jeweils eine Teilschaltung abgespeichert wird. Für Hardwaremakros, Softwaremakros und die eigentlichen Anwenderteilschaltungen existieren 3 Directories. Zur weiteren Bearbeitung der Schaltungen in ARCHIMEDES bzw. in PC-GAD können NBS-Texte erzeugt werden. Um auch Schaltungen grafisch bearbeiten zu können, die früher in NBS erfaßt wurden, besteht die Möglichkeit, aus NBS-Texten Schaltpläne zu generieren. Dabei wird eine automatische Plazierung von Schaltelementen sowie eine automatische Trassierung der Verbindungsleitungen ausgeführt.

• *Fenstertechnik*
Schaltpläne und Symbole für Teilschaltungen sowie Simulationsergebnisse werden auf Arbeitsflächen dargestellt, eingegeben und bearbeitet. Die Anzahl der Arbeitsflächen wird durch die Geräteeigenschaften begrenzt. Auf dem K 8919.11 sind bis zu 30 Arbeitsflächen parallel darstellbar, auf dem PC gibt es keine unmittelbare Beschränkung der Anzahl. Arbeitsflächen können beliebig erzeugt, übereinandergelegt, neu positioniert, verkleinert oder vergrößert und wieder gelöscht werden. Insbesondere ist ein unkomplizierter Übergang von der Bearbeitung einer Arbeitsfläche auf die Bearbeitung einer anderen möglich. (Zum Beispiel Bearbeitung von Schaltungen, die auf mehrere Blätter aufgeteilt sind. Siehe hierzu die Abbildungen.)



Bearbeitung einer Teilschaltung im Gate-Array-Entwurf mit dem grafischen Schaltungseditor TEXgraf



Die Blättertechnik des grafischen Schaltungseditors TEXgraf am Beispiel der Symbolbearbeitung mit hinterlegtem „Innenleben" des Symbols

• *„Einknopfbedienung" mit minimierter Anzahl von Eingabeaktionen*
Alle Funktionen werden mittels grafischer Eingabe (Maus, Tablett) aus Menüs gewählt. Dabei kann in jeder Phase auf eine beliebige andere Menüfunktion zugegriffen werden. Ebenso erfolgen alle Eingaben von Koordinaten (Positionierung bzw. Identifikation von Objekten, Definition von Arbeitsflächen usw.) und die Auswahl von Elementen aus Textlisten grafisch. Lediglich Bezeichnungen für neu erzeugte Elemente sind über die Tastatur einzugeben.

• *Abruf von Funktionserläuterungen zu jedem Zeitpunkt*
Zur Unterstützung der Einarbeitung in das System können unkompliziert Erläuterungen über jede Funktion des aktuellen Menüs abgerufen werden.

• *Steuerung durch Flags*
Die konkrete Zuordnung von Directories, die Verwendung von bestimmten Farbindizes für die jeweiligen Objekte (z. B. Verbindungsleitungen), die Zuordnung von Farbmischungen zu Farbindizes sowie weitere Vorzugseinstellungen werden durch Flags gesteuert. Diese werden in der Initialisierungsphase aus Konfigurationsdateien gelesen (System-, Gruppen- und Nutzerebene), können aber auch interaktiv verändert werden.

Für die Entwicklung von TEXgraf erwies sich die Verwendung der Programmiersprache C als recht vorteilhaft. Unter anderem konnten durch Nutzung der bedingten Kompilierung alle angegebenen Versionen aus einem Paket von Quelldateien gewonnen werden. Ein Auseinanderlaufen des Projektes wird dadurch vermieden. Bei Einhaltung des ANSI-Sprachstandards gab es keine grundlegenden Portierungsprobleme. Schwierigkeiten bereiteten teilweise die völlig verschiedenen Grafikkonzepte von K 8919.11 und PC sowie die Unterschiede bei der Verwendung von betriebssystemabhängigen Funktionen (Dateiarbeit).

Kontaktadresse:
VEB Textimaelektronik Karl-Marx-Stadt
Schaltkreiszentrum
Max-Türpe-Str. 37, Karl-Marx-Stadt, 9043; Tel. 27 21 18

ISSN 0232-2892
Mikroprozessortechnik, Berlin 4 (1990) 6
S. 161–192

vorgestellt

EC 1835

„Tempomacher – schneller als sein Vorgänger". Mit diesem Slogan präsentierte das Kombinat Robotron zur diesjährigen Leipziger Frühjahrsmesse seinen neuesten 16-Bit-Personalcomputer. Der EC 1835 – nach eigenen Angaben „funktionell kompatibel zum Industriestandard PC/AT-03" – basiert auf dem Mikroprozessor U 80601 aus dem Kombinat Mikroelektronik, der bereits im Vorjahr auf der Leipziger Frühjahrsmesse zu sehen war.

Damit ist es Robotron gelungen, rund 5 Jahre nach der Vorstellung des IBM PC/AT einen vergleichbaren Rechner anzubieten. Der Hinweis auf die Kompatibilität des Prozessors läßt auf Kompatibilität zum Intel 80286 des Original-PC/AT schließen, die Aussage, daß „der *modernste* Stand der Mikroelektronik durch Nutzung eines leistungsfähigen Mikroprozessors" angewendet wurde, träfe jedoch ebenso auf einen 80386 oder 80486 zu. Aufschluß darüber kann die Leistungsfähigkeit eines Rechners bringen. Versucht man deshalb in Erfahrung zu bringen, *wie* schnell der EC 1835 nun eigentlich ist, so erfährt man keine MIPS oder FLOPS, sondern, daß er 3- bis 8mal schneller als sein Vorgänger ist.

Besagter Vorgänger ist der zwei Jahre ältere EC 1834, dessen neue Version EC 1834.01 ebenfalls zur LFM '90 Premiere hatte. Der EC 1834.01 erreicht – im Unterschied zum EC 1834 – „die direkte XT-Kompatibilität". Auf diesen Rechner werden wir ebenso wie auf Robotrons ersten 32-Bit-PC, den EC 1835 turbo, und die Transputer-Workstation in unserem Bericht von der Leipziger Frühjahrsmesse (in MP 7/90) näher eingehen.

Zurück zum 1835: Der PC wird standardmäßig mit einer Taktfrequenz von 8 MHz angeboten; das Gerät auf unserem Foto hatte sogar 10 MHz zu bieten. Auf der Systemplatine befindet sich neben der CPU unter anderem ein Koprozessorsockel (Angaben zum Koprozessortyp werden nicht gemacht), der RAM mit 1, 2, 4 oder 8 MByte (über Module steckbar), der ROM mit 16 oder 128 KByte sowie ein Tastatur- und ein Mausanschluß. Die Tastatur ist die gleiche wie beim EC 1834. Als Massenspeicher dienen ein 5,25-Zoll-Festplattenlaufwerk mit 20 oder 42 MByte Kapazität. Mit Slimline-Laufwerken und den entsprechenden Adapterkarten soll die Festplattenkapazität auf maximal 140 MByte ausgebaut werden können. Weiterhin verfügt der EC 1835 über ein bis zwei Floppylaufwerke (5,25 Zoll: 1,2 MByte oder 3,5 Zoll: 1,44 MByte). Insgesamt finden fünf Laufwerke im Gehäuse Platz.

Im Unterschied zum EC 1834 wurde für die Erweiterungskarten von indirekten zu direkten Steckverbindern übergegangen. Damit können nun auch andere IBM-kompatible Erweiterungskarten verwendet werden – vorausgesetzt natürlich, daß sie von der Software unterstützt werden.

Einige technische Angaben

Prozessor	U 80601
optional	Koprozessor
Taktfrequenz	8 MHz
Hauptspeicher	1 MByte, erweiterbar auf der Hauptplatine auf 8 MByte
Floppy-Disks	1 bis 2 5,25 Zoll: 1,2 MByte/3,5 Zoll: 1,44 MByte
Harddisk	5,25 Zoll: 20 der 42 MByte, 5 MBit/s
Grafik	MGA: 720 × 348 Pixel
optional	VGA: 640 × 480 Pixel, 256 Farben
Monitor	monochrom, 31 cm
optional	color, 38 cm, 16 Farben
Uhr	batteriegepuffert, CMOS-RAM
Steckplätze	6 × 16 Bit, 2 × 8 Bit (4 belegt durch Grafik-, Floppy-, Harddisk- und I/O-Adapter)
Erweiterungs-karten	-I/O-Adapter (1 × Centronics, 2 × V.24 oder IFSS) -X.21-/X.25-Adapter (für paketvermitteltes Datennetz) -BSC-Adapter (1 × synchrone oder asynchrone V.24) -LNC2-Adapter für Vernetzung im Rolanet-2/Ethernet oder im Thin-Ethernet -IEC-Bus-Adapter -KIF-Adapter (Anschluß an Bildschirmsystem EC 7920)
Betriebssysteme	DCP 3.30, Mutos 1835
Software	-Basic-Interpreter und -Compiler, T-Basic, T-Pascal, C, Compiler für Fortran, Cobol, Modula, Interpreter für Lisp, Prolog -TP, Multicalc II, Aidos, Redabas 4 und Redacom -CAD-Pakete wie Multicad, Proconv, LPCAD, GKS 16, GKS 1800, Gramos, Autotech und PCLES -LAN-Software: SCOM-LAN, DCSDCP, EC-NET, EC-LAN, Prolan -mathematische Pakete und integrierte Software, z. B. Ariadne und Multicomp

Neben den auf der Systemplatine vorhandenen Baugruppen sind für eine Grundausstattung von den 8 Steckplätzen bereits 4 für die Grafikkarte (Hercules oder VGA), für den Floppycontroller, den Harddiskcontroller sowie für eine I/O-Karte mit paralleler und serieller Schnittstelle (beispielsweise für einen Drucker) vergeben.

Positiv hervorzuheben sind die nun zur Standardausrüstung gehörende batteriegepufferte Uhr und der Paßwortschutz. Der in der Grundausstattung vorhandene 1-MByte-RAM erlaubt das Anlegen eines Shadow-RAMs, mit dem die BIOS-Routinen beschleunigt werden können.

Erweiterungsmöglichkeiten bietet Robotron mit einer Reihe von Adapterkarten. So kann die vorhandene – wie jede weitere – Harddisk-Controllerkarte zwei Festplatten bedienen; das gleiche gilt für die Floppy-Karte. Der Monochrom-Grafik-Adapter (MGA) kann durch einen Adapter mit Video Graphics Array (VGA) ausgetauscht werden. Über den IEC-Bus-Adapter ist eine Steuerung von bis zu 16 Geräten in einem Meßsystem möglich, und an den I/O-Adapter mit einer parallelen Schnittstelle (V.24 oder IFSS) können Drucker, Plotter, grafisches Tablett oder Modem angeschlossen werden. Der BSC-Adapter dient zur Kopplung mit Rechnern der ESER-Klasse.

Die Möglichkeit der Vernetzung der Rechner in einem LAN ist international bereits zum Standard geworden. Dem trägt Robotron mit dem Angebot des LAN-Adapters LNC2 (Lokalnetzcontroller) Rechnung. Er erlaubt im Ethernet-kompatiblen Rolanet-2 eine Übertragungsgeschwindigkeit von 10 MBit pro Sekunde. Der LAN-Adapter enthält neben der Schnittstelle zum Transceiver des Rolanet-2 noch einen On-board-Transceiver mit einem Koaxialanschluß für ein Thin-Ethernet. Dieses Thin-Ethernet ermöglicht die Verkopplung von 30 Rechnern bei einer maximalen Segmentlänge von 180 Metern. Dagegen bietet das vollständige Rolanet-2 100 Rechnern eine maximale Segmentlänge von 500 Metern, hier werden aber die zusätzlichen Transceiver-Baugruppen benötigt.

Robotron bietet zwei Betriebssysteme für den EC 1835 an: DCP 3.30 (DCP – disc control program) und Mutos 1835. Das vom EC 1834 her bekannte „DCP ist durchgängig kompatibel zum international weitverbreiteten System PC-DOS", wobei unklar bleibt, ob die Versionsnummern übereinstimmen. Das DCP wird, wie auch einige Tools und Programmiersprachen, „gebündelt" mit der Hardware vertrieben. Obwohl Robotron den Vorteil des einheitlichen Betriebssystems für EC 1834 und 1835 hervorhebt, werden für den EC 1835 zusätzlich angeboten: Lisp- und Prolog-Interpreter, Redabas-4. Das Unix-kompatible Mutos 1835 kann „solo" bezogen werden. *MP-Hk*

ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

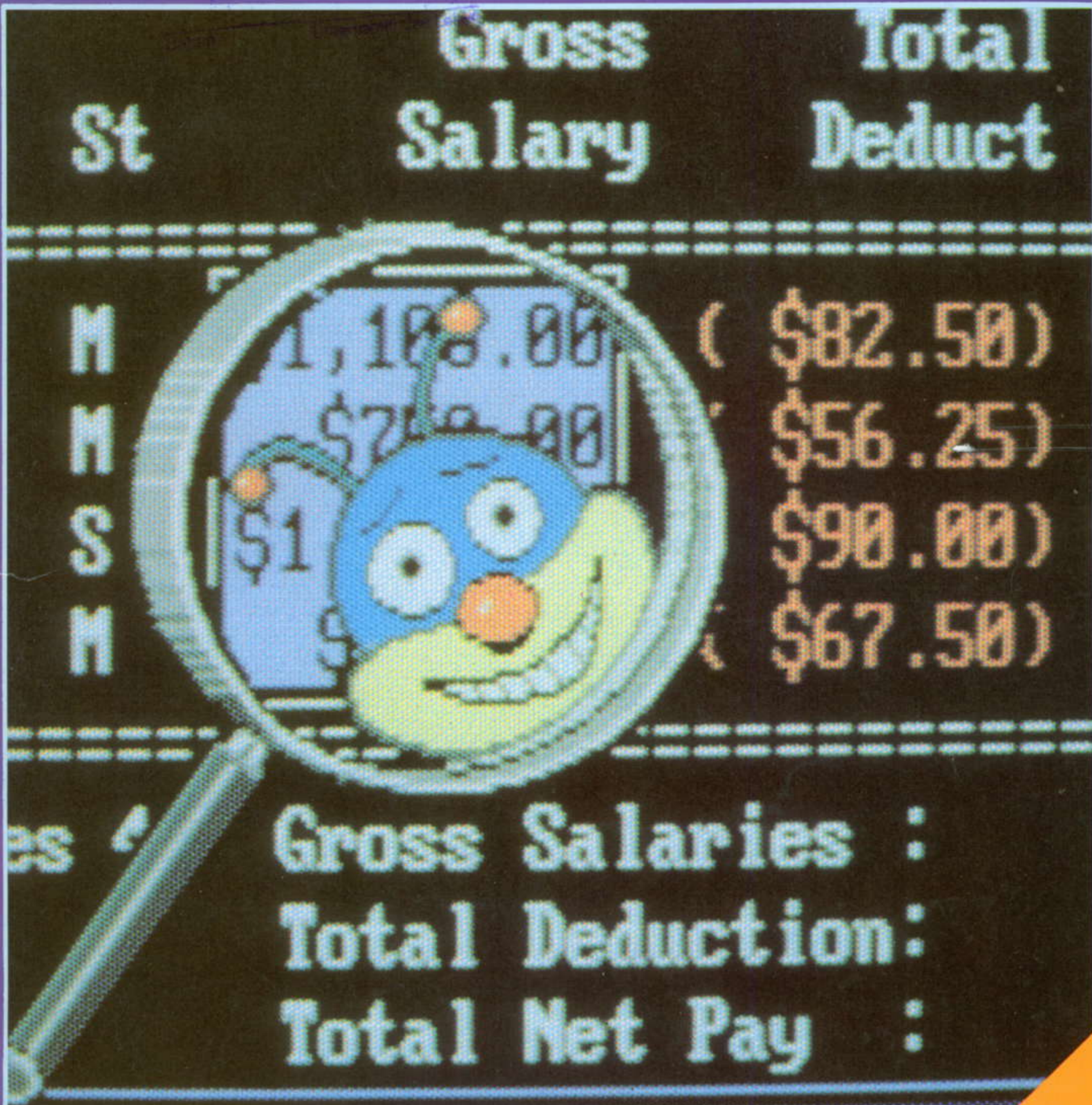
MP

Heft 7 · 1990

Mikroprozessortechnik

Verlag Technik Berlin
ISSN 0232-2892
DM 5,–

mit Beiträgen aus c't



Debugger

Wie man Störenfriede aufspürt

Neuer Kurs Lisp

INTERDECK '90

Nach dem Erfolg der INTERDECK'89 fand in diesem Jahr in Würzburg vom 13. bis 15. März parallel zum 13. Symposium von DECUS München e.V. die zweite unabhängige Messe für VAX- und PDP-11-Anwender statt. Auf einer Fläche von über 1000 qm stellten über 70 Firmen ergänzende Hardware, kompatible Peripheriegeräte, Netzwerkprodukte, Softwareprodukte und komplette Systemlösungen für Rechenanlagen der Digital Equipment Corporation (DEC) aus. Organisiert wurde die Messe von der Network GmbH unter der Schirmherrschaft der unabhängigen Zeitschrift DECKBLATT für VAX- und PDP-11-Anwender vom Verlag Markt & Technik. Obwohl eine Reihe führender Firmen wie Emulex und COMPAQ dieses Jahr nicht vertreten waren, zählte die Messe wieder über 2000 Besucher und wurde von den ausstellenden Firmen vor allem wegen der Nähe des DECUS-Symposiums wieder als erfolgreich eingeschätzt.

Schwerpunkte auf der Messe waren bei einer Vielzahl von Firmen vor allem Speichersubsysteme, Backup-Geräte und Kommunikationsprodukte.

Mit einem umfassenden Angebot an DEC- und DEC-kompatibler Hardware, PCs von COMPAQ und Netzwerkprodukten einschließlich der erforderlichen Lizenzen war die Fa. pdv-systeme GmbH Goslar vertreten. Nachdem bereits im vergangenen Jahr ein Kassettenmagnetbandspeicher nach dem VHS-Verfahren mit 2,5 GByte Speicherkapazität angeboten wurde, stellte die Firma dieses Jahr das Kassettenmagnetbandgerät GIGASTORE-54 (ca. 17500 DM) von der DIGI-DATA-CORPORATION mit einer Kapazität von 5,4 GByte pro Kassette aus (Bild 1). Der Datenträger ist eine Digital-Kassette analog einer E-180-Video-Kassette.

Von der FORUM Technische Entwicklung, Vertrieb und Service GmbH München-Olching wurde das bereits letztes Jahr angekündigte löschbare optische Plattenspeichersystem INSPIRE nach der bewährten Technologie von SONY und 3M angeboten. Das Laufwerk IDQ-10 mit einer Speicherkapazität von 630 MByte kostet ca. 14000 DM und das Doppellaufwerk IDQ-20 mit einer Kapazität von 1,3 GByte ca. 25000 DM. Die $5\frac{1}{4}$-Zoll-CD ist mit 650 DM noch relativ teuer. Für DEC-Rechner gibt es ein Q-Bus-, UNIBUS- und MSCP-Interface. Unterstützt wird das Laufwerk von VAX/VMS, ULTRIX und RSX-11M. Die durchschnittliche Zugriffszeit beträgt ca. 83 ms und die maximale Datenübertragungsrate 7 MBit/s.

Auf dem Gebiet der Netzwerkprodukte sollen vor allem die Terminalserver MAXserver 1500 und MAXserver 1800 mit Unterstützung des LAT- und TCP/IP-Protokolls in gemischten Sitzungen erwähnt werden (Bild 2). Beide Terminalserver unterstützen 16 Terminals mit einer Datenübertragungsrate von 300 Baud bis 38,4 kBaud sowie einen wahlfreien Anschluß an Thick- oder Thin-Wire-Ethernet. Der MAXserver 1800 hat ein $3\frac{1}{2}$-Zoll-Floppy-Laufwerk zum Laden der Terminalserversoftware und ist damit von einem Host unabhängig. Die Preise betragen ca. 7200 DM bzw. 8400 DM. Auf der Grundlage der 8-mm-Video-Kassette war weiterhin das Kassetten-Backup-Laufwerk GT88 von MegaTape mit einer Kapazität von 2,3 GByte/Kassette (ca. 11000 DM) im Angebot.

Von der transtec GmbH Tübingen, einem der größten autorisierten Händler für DEC- und DEC-kompatible Produkte in der Bundesrepublik, wurden zahlreiche neue Produkte wie X-Window-Terminals, VT-300-kompatible Terminals von Microterm, SCSI-Plattensubsysteme, DSSI-kompatible Plattenlaufwerke, DSA-kompatible Platten- und Magnetbandlaufwerke, Terminalserver von LANtronix, Magnetbandkassettensubsysteme, u. v. a. m. angeboten. Besonders herausgegriffen werden sollen hier der LANtronix-Terminalserver ETS-8 (Bild 3), die Ethernet-Doppel- und -Vierfachtransceiver GCI-2×0/GCI-4×0 (Bild 4) und der postscriptfähige Vierfarb-Thermotransfer-Drucker Color Script 100 von QMS (Bild 5). Der ETS-8 (ca. 3000 DM) unterstützt 8 Terminals mit je max. 8 Sitzungen nach dem LAT- oder TCP/IP-Protokoll.

Die Transceiver GCI-200 und GCI-400 gestatten den Anschluß von zwei bzw. vier Stationen an ein Thick-Wire-Ethernet-Segment (600 bzw. 1000 DM). Sie sind vor allem dort sinnvoll einsetzbar, wo der minimale Abstand von 2,5 m zwischen den Einzel-Transceivern nicht eingehalten werden kann. GCI-210 und GCI-410 sind für Mininetzwerke ohne Ethernet-Segment gedacht, während GCI-220 und GCI-420 den Anschluß an einen externen Transceiver zulassen.

Der Vierfarb-Thermotransferdrucker Color Script CS100 (ca. 3400 DM) basiert auf einem Mitsubishi-Druckwerk und einem

Fortsetzung siehe 3. Umschlagseite

Bild 1. Kassettenmagnetbandspeicher GIGASTORE-54 nach dem VHS-Verfahren mit 5,4 GByte/Kassette von der DIGI-DATA-CORPORATION

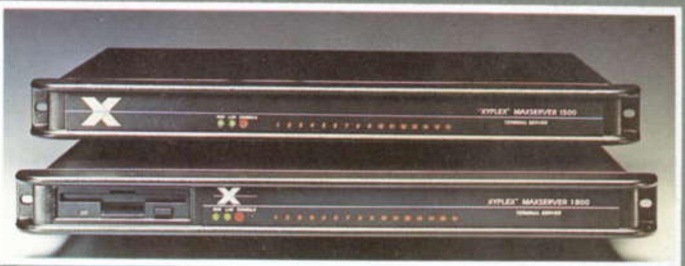


Bild 2. Terminalserver MAXserver 1500 bzw. MAXserver 1800 für max. 16 Terminals oder Drucker mit LAT- und Telnet-(TCP/IP)-Protokoll von XYPLEX

Bild 3. Lantronix-Terminalserver ETS-8 für max. 8 Terminals oder Drucker mit LAT- und Telnet-(TCP/IP)-Protokoll

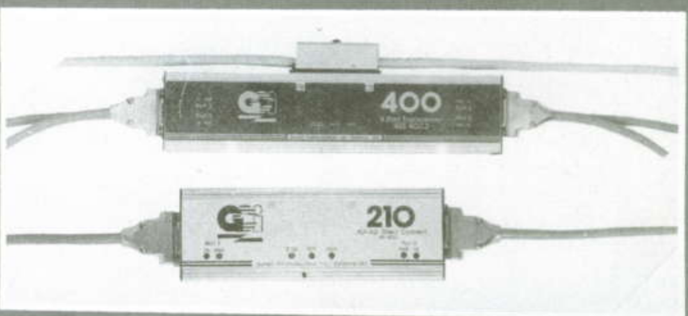


Bild 4. Ethernet-Doppel-Transceiver GCI-210 und Ethernet-Vierfach-Transceiver GCI-400

DEBUG – Eine Bedienoberfläche für 16-Bit-Single-Board-Computer

DEBUG wurde unter DCP auf dem A 7150 zu Inbetriebnahme und Service von Single Board Computern (SBC) entwickelt. Das Programm kommuniziert mit dem Monitorprogramm des SBC über eine serielle Schnittstelle (IFSS) und bietet dem Nutzer durch die Verwendung von Fenstertechnik und Pull-Down-Menüs einen guten Bedienkomfort.

Zum Funktionsspektrum gehören u. a.:

- gleichzeitige Darstellung von Registerinhalten, reassembliertem Code und Daten im HEX- und ASCII-Format
- Unterstützung des Schrittbetriebes
- Locator zum Binden von EXE-Dateien auf Festadressen
- Umwandlung von HEX- in ASCII-Dateien
- Kommandoversion zur direkten Eingabe von Monitor-Kommandos
- Directoryanzeige, Subdirectorywechsel.

Als Entwicklungssystem wurde an Stelle des SBC eine A 7100 genutzt, welcher ohne Hardwareänderungen angeschlossen werden kann.
Der Datenaustausch erfolgt über dessen Tastaturschnittstelle, wobei die Standard-Consolenausgabe auf diese geleitet wird.

Interessenten am Erfahrungsaustausch oder an der Nutzung wenden sich an die:

AdW der DDR,
VDE Potsdam, Bereich wissenschaftlicher Gerätebau
Telegrafenberg A 31, **Potsdam, 1561**, Tel.: 31 02 40

H. Schröter

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag Verlag Technik, Oranienburger Str. 13/14, DDR – 1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Hans Weiß, Chefredakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2870203), Hans-Joachim Hill (Tel. 2870209); Sekretariat: Hannelore Kulik (Tel. 2870381)

Gestaltung Christina Bauer

Titelfoto Hans-Joachim Hill

Beirat Prof. Dr. sc. Thomas Horn, Prof. Dr. Bernd Junghans, Dr. Dietmar Keller, Prof. Dr. sc. Bernd-Georg Münzer, Prof. Dr. sc. Peter Neubert, Prof. Dr. sc. Dr. Michael Roth (Vorsitzender), Dr. Gerhard Schulze, Prof. Dr. sc. Manfred Seifart, Dr. Dieter Simon, Prof. Dr. sc. Dr. Jürgen Zaremba

Lizenz-Nr. 1710 des Presse- und Informationsdienstes der Regierung der DDR

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

Anzeigenannahme Verlag Technik, Anzeigenverwaltung, PSF 201, Berlin, 1020; Tel. 2870254, 2870309

Redaktionsschluß 11. Mai 1990

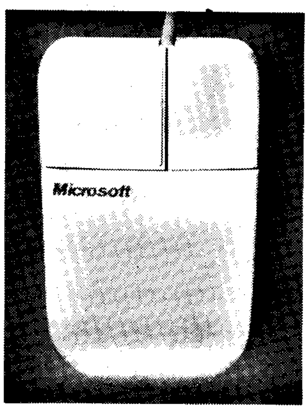
AN (EDV) 49837

Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– DM, Abonnementspreis vierteljährlich 15,– DM; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
Verlag Technik, Abteilung Absatz, PSF 201, Berlin, 1020 oder: *DDR:* sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSFR:* PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *Republik Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Republik Ungarn:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nôi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR – 7010 Leipzig und Leipzig Book Service, Talstraße 29, DDR – 7010 Leipzig

Zum Titelbild
Debugger sind Programme, mit denen man Fehler in anderen Programmen, aber auch Hardwarefehler suchen kann. Das englische Wort *bug* bedeutet Wanze. Die auf dem Titelbild dargestellte Lupe – übrigens aus einem Demoprogramm von SuperCalc 5 der Firma Computer Associates – ist in diesem Fall unser Debugger. Der Beitrag „Debugger – Werkzeug für schlechte Programmierer?" auf der **Seite 195** beschäftigt sich mit dem Debuggen von Programmen und den zu diesem Zweck kommerziell zur Verfügung stehenden Werkzeugen.



Mäuse!
Der heimischen Haus-, Wald-, Brand- und Zwergmaus gesellt sich seit einigen Jahren die Digitalmaus hinzu, als eher nützliches Anhängsel am Computergehäuse.
Der aus der c't übernommene Beitrag „Digitale Nager" auf der **Seite 203** beleuchtet, was sich von der Mausbewegung über das elektrische Signal bis hin zur Anwendung abspielt.

Vorschau
Für Heft 8/1990 bereiten wir für Sie Beiträge zu folgenden Themen vor:
- Digitale Signalprozessoren
- Peripherieschaltkreise der Reihen DL 8600 und DS 8600
- Rasterbilder auf dem PC

Inhalt

MP-Info 194

Uwe Schulze
Debugger – Werkzeug für schlechte Programmierer? 195

Dr. Christian Hanisch
Execlook registriert Programmaufrufe 198

Jörg Moebes, Dr. Veronika Sauer, Bernd Däne, Lutz Finn
Prüfmuster für Register- und Busstrukturen 201

Dieter Michel
Digitale Nager 203

kennengelernt
JPI TopSpeed Modula-2 206

MP-Kurs 207
Volker Bauche, Horst Friedrich
Lisp (Teil 1)

Wegbereiter der Informatik
Vannevar Bush 211

MP-Computer-Club 212
Berthold Biener:
Packer oder Die Kompression von Daten

Entwicklungen und Tendenzen 214

MP-Börse 215

MP-Literatur 216

MP-Leserumfrage 217

MP-Bericht 218
Leipziger Frühjahrsmesse 1990

Verband der Computerclubs gegründet

Computerfreaks, die bisher im Radiosportverband der DDR organisiert waren, haben einen selbständigen Verband der Computerclubs (VCC) gegründet.
Hauptziel des Verbandes ist die Förderung und Verbreitung der Hobbyanwendung von Computertechnik und Software, soweit sich diese mit den moralischen, ethischen und juristischen Normen in Einklang befinden. Im Verband kann jeder Mitglied werden. Die Mitgliedschaft wird an Einzelpersonen vergeben, wobei ein Club als Mitglied des VCC registriert wird, wenn ihm mindestens fünf VCC-Mitglieder angehören. Die Eigenständigkeit der registrierten Computerclubs im Verband bleibt erhalten. Die kooperative Zusammenarbeit mit anderen bereits aktiven Gruppen und Einrichtungen wird angestrebt.
Das Kernstück der Verbandsarbeit soll vor allem das Gebiet der Leistungen für Verbandsmitglieder werden, die für Einzelnutzer oder Clubs zu uneffektiv sind. Der Leistungsumfang soll abhängig von den Möglichkeiten u. a. die Gebiete Erfahrungsaustausch, Bildung, Konsultation, Softwareinfo, Dokumentationsbereitstellung, Ausstellungen und Leistungsvergleiche enthalten. Die Leistung des Verbandes wird anfangs vor allem in der Vermittlung der entsprechenden Kontaktadressen bestehen. Der Ausbau des Leistungsumfanges wird durch die Verbandsmitglieder in Form von Angebot und Nachfrage mitbestimmt.

Hinweise, Meinungsäußerungen und Anfragen können an folgende Adresse gerichtet werden:
Verband der Computerclubs, Geschäftsstelle, Langenbeckstr. 36–39, Neuenhagen, 1272

Ein Rechnerpionier wurde 80

Professor Dr. mult. Konrad Zuse – der große Pionier der Computertechnik – beging im Juni dieses Jahres seinen 80. Geburtstag. Zuse ist der Erfinder und Erbauer des ersten voll funktionstüchtigen, programmgesteuerten digitalen Rechenautomaten der Welt. Für seinen 1941 fertiggestellten Automaten Z3 hatte Zuse folgende Neuerungen erstmals in den Rechenanlagenbau eingebracht:
- die Verwendung des Dualsystems anstelle des traditionellen dekadischen Zahlensystems
- die Darstellung der Zahlen in Gleitkommaform
- die Einführung des Prinzips der Programmierung mit vercodeten Adressen.

Mit den Entwicklungsarbeiten hatte Zuse bereits im Jahre 1935 begonnen; ein erstes Ergebnis war der 1938 fertiggestellte Automat Z1, der bereits alle wesentlichen Funktionseigenschaften des Z3 besaß, jedoch mit rein mechanischen Speichern ausgelegt war. Der Z3 war dann mit Relaisspeichern ausgerüstet. Beide Maschinen sind im Krieg leider zerstört worden, doch wurde der Z3 nach dem Krieg noch einmal nachgebaut. Obwohl auch die Unterlagen dazu vernichtet waren, gelang Zuse in den letzten Jahren auch der nochmalige Nachbau seines allerersten Automaten Z1, der heute im Berliner Museum für Verkehr und Technik bewundert werden kann. *B.*

Eine GOLDENE DISKETTE erstmals in die DDR

Bundesforschungsminister Dr. Heinz Riesenhuber hat auch dieses Jahr wieder drei Nachwuchs-Programmierer mit der GOLDENEN DISKETTE ausgezeichnet. Sie sind Sieger in einem alljährlich vom Mikrocomputer-Magazin CHIP zusammen mit dem Informationszentrum „Jugend und Technik" der Hannover Messe veranstalteten Programmier-Wettbewerb. Im siebten Jahr der Veranstaltung sind insgesamt 341 Beiträge eingesandt worden, darunter 32 Programme von Teilnehmerinnen. Gesucht wurden Lernprogramme, die auf unterhaltsame Art und Weise in ein Wissensgebiet einführen.
Erstmals konnten auch junge Programmier-Talente aus der DDR an dem Wettbewerb teilnehmen. Und Ulf Ackermann, dem 25jährigen Physik- und Astronomie-Studenten aus Erfurt, glückte gleich beim ersten Mal der Sprung unter die ersten Drei. Sein Programm „Das Planetarium im Computer" hat er auf einem Schneider PC 1512 DD geschrieben. *MP*

OMRON kooperiert mit Carl Zeiss JENA

Das weltweit operierende Elektronikunternehmen OMRON Corporation gab anläßlich der Leipziger Frühjahrsmesse seine zukünftige Zusammenarbeit mit Carl Zeiss JENA bekannt. Die OMRON Corporation zählt nach eigenen Angaben in Japan zu den Marktführern auf dem Gebiet elektronischer Systeme und Komponenten für die Automation und ist unter den weltweit 500 größten Industrieunternehmen. Die Verbindung von OMRON mit Carl Zeiss JENA wurde über die westdeutschen Niederlassungen Omron Electronics GmbH, Düsseldorf und die Omron Geschäftssysteme GmbH, Hamburg aufgenommen.
Grundlage dieser Zusammenarbeit ist der Vertrieb von Komponenten und Systemen unter anderem für die Fabrikautomation. *MP*

Siemens-Schulung in der DDR

Nach der Gründung eines gemeinsamen Software- und Systemhauses „Siemens-SAP-Robotron" während der Leipziger Messe wurde jetzt die Eröffnung einer Siemens-Schule für Daten- und Informationstechnik in Dresden als Ergänzung auf dem Schulungssektor vereinbart. 30 Robotron-Mitarbeiter absolvierten im April eine neuntägige Fortbildung in der Anwendung der Rechnersysteme BS2000 und SINIX/UNIX. Es ist geplant, im Laufe des Sommers diese Aktivitäten zu verstärken und eine eigene Siemens-Schule für Daten- und Informationstechnik in Dresden aufzubauen. Sie soll zum Jahresende 12 Schulungsräume mit 15 eigenen Dozenten umfassen. Neben Mitarbeitern von Siemens-SAP-Robotron werden auch andere Interessenten aus der DDR an den Kursen zum DI-Weiterbildungsprogramm einschließlich der Managerweiterbildung teilnehmen können. *MP*

Taylorix-Beratung in der DDR

Um dem Bedarf an marktwirtschaftlichem Basiswissen und an den dazugehörenden Organisationslösungen gerecht werden zu können, plant die Taylorix AG in Zusammenarbeit mit örtlichen Partnern den Aufbau einer Beratungs-, Schulungs- und Dienstleistungsorganisation in sechs Städten der DDR.
Außer den Computerlösungen der Taylorix AG sollen die Taylorix-Partner in der DDR besonders die Beratungsleistungen der Taylorix-Beratungs-GmbH sowie Aus- und Weiterbildungskurse des Taylorix-Institutes für Bildung und Beratung GmbH für das EDV-gestützte Rechnungswesen anbieten.

Kontaktadresse: Taylorix AG, Postfach 40 08 67, D-7000 Stuttgart 40, Tel. 87 07-3 81 *MP*

Nantucket News

Das Supportmagazin der Firma Nantucket für Clipper und die Nantucket Tools II ist ab sofort für jedermann erhältlich, unabhängig davon, ob der Interessent bereits ein Nantucket-Produkt besitzt. Die Nantucket News erscheint vierteljährlich und beschäftigt sich mit allem, was es zu und rund um Clipper gibt. Der direkte Kontakt des Nantucket-Supports mit den Verfassern der Artikel gibt dem Magazin die Praxisnähe, die für Clipper-Entwickler wichtig ist.
Die Zeitschrift kann zu einem Preis von DM 49,00 plus 14 % Mehrwertsteuer für das Jahresabonnement unter folgender Anschrift bezogen werden:
Nantucket GmbH, Mülheimer Straße 79, D-5090 Leverkusen. *MP*

Rank Xerox Deutschland jetzt für DDR zuständig

Rank Xerox, bisher in der DDR vertreten durch eine Repräsentanz der East European Organisation der Rank Xerox Ltd., London, überträgt die Verantwortung für den Auf- und Ausbau eines eigenständigen Vertriebs- und Kundendienstnetzes an das Management der Rank Xerox GmbH, Düsseldorf. Vorrangiges Ziel des deutschen Managements ist es, für alle Kunden auf dem Gebiet der DDR einen flächendeckenden Service sicherzustellen, der dem Qualitätsstandard von Rank Xerox entspricht. Zu diesem Zweck wird die Gründung einer Vertriebs- und Wartungsgesellschaft, gegebenenfalls zusammen mit einem eingeführten lokalen Partner, angestrebt. *MP*

Unerwartete CADdy-Anwenderzahl

Wie der Mönchengladbacher CAD-Hersteller ZIEGLER-Instruments mitteilt, ist sein System CADdy in der DDR mehr als 3000mal installiert. Über diesen hohen Verbreitungsgrad zeigte sich das Software-Haus einigermaßen überrascht. Zwar hatten bereits vor Öffnung der Mauer intensive Ostkontakte bestanden, und Software-Verkäufe in die DDR waren an der Tagesordnung. Jetzt zeigte sich allerdings, daß man die Rechnung mit den Verkaufszahlen ohne die illegal verbreiteten Pakete gemacht hatte, die auf einer älteren CADdy-Version basieren, aber sonst dem Gegenstück bis aufs Bit gleichen, einmal abgesehen von einigen kosmetischen Veränderungen der Benutzeroberfläche.
Diese Software kann jetzt in Originale verwandelt werden: DDR-Anwendern mit Plagiat-Versionen bietet ZIEGLER-Instruments die Möglichkeit zum Umsteigen auf die aktuelle und offiziell lizensierte CADdy Version zu günstigen Sonderkonditionen. Damit kommt der Nutzer auch in den Genuß der Service- und Supportleistungen des Herstellers.

Kontakt: ZIEGLER-Instruments GmbH, Geschäftsbereich CADdy, Nobelstr. 5, D-4050 Mönchengladbach 4.

ABACOMP-Filialen in der DDR

Von bisher acht Filialen werden seit einigen Wochen in der DDR vorwiegend Universitäten und Ämter sowie Betriebe des Handels und Handwerks mit Personalcomputern und Peripherie der Firma ABACOMP versorgt. Hauptsächlich ehemalige Chefingenieure sowie anerkannte Hochschulkapazitäten der Fachbereiche EDV und Elektrotechnik bilden die neue Mannschaft. Weitere Filialen sind zwar bereits in Vorbereitung, jedoch werden (insbesondere für den nördlichen Landesteil) gerne noch entsprechend qualifizierte Partner in das Vertriebs- und Servicenetz aufgenommen.

Kontakt: ABACOMP GmbH, Kransberger Weg 24, D-6000 Frankfurt/M. 50; Tel. 76 30 39

Robotron kooperiert

Das Kombinat Robotron gründet gemeinsam mit der BRD-Firma Kienbaum & Partner ein Gemeinschaftsunternehmen.
An der *Unternehmensberatung Kienbaum GmbH* genannten Gesellschaft wird der in Gummersbach ansässige westdeutsche Partner mit 75 Prozent der Anteile die Mehrheit halten. Nach Aussagen von Jochen Kienbaum, Vorsitzender der Geschäftsführung von Kienbaum & Partner, erwartet sein Unternehmen eine Ausweitung des Stammgeschäftes in eine „zukunftsträchtige Wachstumsregion". Zielgruppe seien mittelständische Unternehmen ab 100 Beschäftigten. *MP*

Debugger –

Werkzeug für schlechte Programmierer?

Uwe Schulze, Berlin

Auch wenn die Überlieferung, der Begriff Debugger (bug, englisch: Wanze) rühre von einem Insekt her, das mit seiner Anwesenheit zwischen den Relaiskontakten eines Großrechners diesen außer Betrieb gesetzt hat, wohl nicht authentisch ist, so umreißt der heute ausschließlich auf Software bezogene Begriff doch das Ziel: Die Beseitigung heimtückischer, schwer zu lokalisierender, logischer Fehler.

„Richtige“ Programmierer sind zwar häufig der Meinung, keine Fehler zu begehen oder diese durch logische Überlegung zu finden, nichtsdestotrotz kann ein Debugger ein sehr effizientes Werkzeug sein, um komplizierten Fehlern schnell auf den Grund zu gehen. Denken Sie nur an Assemblerprogramme oder die Nebeneffekte in Hochsprachen. Das darf natürlich nicht darüber hinwegtäuschen, daß Fehlern am besten vorgebeugt wird, wenn gut strukturiert nach einem sorgfältigen Entwurf Modul für Modul programmiert und getestet wird.

Die Idee des Debugging besteht in der Beobachtung des Programms während der Abarbeitung. Diese Überwachung bezieht sich auf den Programmablauf (Sprünge, Unterprogrammrufe) und die aktuelle Belegung der Daten. Es ist leicht zu verstehen, daß das für eine Hochsprache recht kompliziert ist und wegen der unterschiedlichen internen Darstellung für jeden Compiler ein passender Debugger benötigt wird. Nur ein Beispiel ist die unterschiedliche Realzahl-Darstellung.

Debuggen auf Maschinenebene

Die ersten PC-Debugger waren wenig komfortabel und gestatteten nur die Arbeit auf Maschinenebene. Deshalb dienten sie auch in erster Linie zur Unterstützung von Assemblerentwicklungen. Der Standarddebugger für CP/M war ZSID (Zilog Symbolic Instruction Debugger), zum Lieferumfang von MS-/PC-DOS (kurz: DOS) gehört das Programm Debug. Im Mittelpunkt der Arbeit eines Debuggers steht der Trace-Befehl. Damit können Programme im Einzelschrittbetrieb abgearbeitet werden. Mit

t = 100 5

werden mit Debug 5 Assemblerbefehle ab der Adresse 100 ausgeführt und jeweils die Register angezeigt, wodurch der Programmverlauf (Verzweigungen und Datenbelegung) überwacht werden kann. Um Schleifen nicht in Einzelschritten durchlaufen zu müssen, gibt es die Möglichkeit, Unterbrechungspunkte (Breakpoints) zu setzen, die das Programm an entsprechender Stelle stoppen. Ein direktes Ansehen von Datenbereichen in hexadezimaler Form kann mit dem Kommando Dump erfolgen, beispielsweise dient

d100 110

zur Anzeige des Speicherbereiches von Adresse 100 bis 110 (bezogen auf das Datensegmentregister, wenn nicht anders angegeben). Um aber die Zuordnung der Programmkonstrukte zu den aktuell belegten Speicherplätzen zu erhalten, benötigt der Programmierer ein List-File, das vom Assembler erzeugt werden kann. Hier werden auch die Grenzen dieser Art von Arbeit sichtbar: Komplexe Programme lassen sich so kaum verfolgen. Eine wertvolle Option des Debuggers ist das Assemblieren. Damit können kurze Assemblerstücke in Assemblermnemonik eingegeben werden. Mehr dazu jedoch später. Ebenso wird das interaktive Reassemblieren unterstützt.

Weitere nützliche Funktionen eines Debuggers sind eine einfache Hexadezimal-Arithmetik (vor allem zur Adreßrechnung), sowie Füll-, Vergleichs-, Verschiebe und Suchoperationen im Speicher.

Symbolisches Debugging

Der nächste Schritt ist die Nutzung von Symbolen des Quelltextes für die Fehlersuche. Bereits der erwähnte Debugger ZSID bietet diese Möglichkeit. Für DOS kann der Standarddebugger Symdeb verwendet werden, der mit dem Assemblerpaket geliefert wird. Der Programmierer wird der Aufgabe enthoben, mit den konkreten Adressen zu operieren. Statt dessen können Label-, Segment-, Prozedur- und Variablennamen direkt angegeben werden. Die Verbindung wird über ein Sym-File hergestellt, das beim Linken angelegt wird. Dazu muß die Assemblierung mit folgender Kommandofolge abgewickelt werden:

① Alle Symbole als public deklarieren
② Assemblieren: masm test,,;
③ Linken und Map-Datei erzeugen: link test,,/map;
④ Erzeugen Symboldatei: mapsym test
⑤ Starten des Debuggers: symdeb tes.sym test.exe

Es können auch mehrere Sym-Dateien geladen werden.

Um sich jetzt einen 16-Bit-Wert anzusehen, der an der Marke Wert steht, reicht der Befehl

dw ds:wert

DW steht für *Dump Word* und DS kennzeichnet das Datensegment. Tafel 1 enthält eine Kommandoübersicht von Symdeb. Die neue Qualität von Symdeb zeigt sich auch in der Nutzung verschiedener Bildschirmseiten für Testprogramm und Debugger (damit die Ausschriften des Debuggers nicht das Bild des Testprogrammes zerstören), Möglichkeiten der getrennten Ein-/Ausgabeumleitung für den Debugger mit < und > und das Testprogramm mit { und } sowie in der Berechnung arithmetischer Ausdrücke.

Praktisch gegenüber Debug ist ferner die Möglichkeit, sich Speicherbereiche in verschiedenen Formaten anzeigen zu lassen (Dump). So kann man sich den Interruptvektor beispielsweise mit DD (für Dump Doubleworld) im Adreßformat ansehen. Der entsprechende Befehl dazu heißt

dd 0:0

und erleichtert die Lesbarkeit der bei Intel-Prozessoren üblichen Highteil-/Lowteil-Darstellung (siehe Bild 1). Die Eingabeumleitung wird oft dazu benutzt, kleine Assemblerprogramme mit dem Debugger zu erstellen, ohne auf einen Editor zu verzichten. Beispielsweise läßt sich folgende Befehlsfolge

Tafel 1 Die Symdeb-Kommandos im Überblick

Kommando	Bedeutung
A[adresse]	Assemblieren
BP[nr] adresse [zaehler] [kommando]	Breakpoint setzen
BC liste | *	Breakpoint löschen
BL	Breakpoints listen
C bereich adresse	Vergleichen von Speicherbereichen
D[typ] [adresse | bereich]	Speicheranzeige
E[typ] adresse [liste | wert]	Tastatureingabe
F bereich liste	Füllen eines Speicherbereiches
G[=startadresse] [haltepunkte]	Starten der Abarbeitung
H wert1 wert2	Einfache Hexadezimal-Rechnung
I port	Lesen des Ports
L[adresse[lw: satz anzahl]]	Laden
M bereich adresse	Verschieben im Speicher
N[datei] [argumente]	Festlegung Dateiname
O port byte	Ausgabe zum Port
P[=startadresse][anzahl]	Einzelschrittbetrieb ganzer Funktionen
Q	Symdeb beenden
R[register[[=]wert]]	Register anzeigen/mit Wert belegen
S bereich liste	Bytefolge Suchen
T[=startadresse] [anzahl]	Einzelschrittbetrieb
U[bereich]	Reassemblieren
W[adresse[lw: satz anzahl]]	Speichern, Schreiben
X?[tabelle] [segment:][symbol]	Anzeigen der Symboltabelle
XO [tabellenname] [segmentname]	Öffnen der Symboltabelle
Z symbol wert	Zuordnung einer Adresse zu einem Symbol
?[ausdruck]	Berechnen eines Ausdrucks ohne Parameter: Hilfe
![kommando]	Ausführung von DOS-Kommandos

Typ:

B	Byte
A	ASCII
W	Wort
S	kurze Realzahlen
L	lange Realzahlen
T	10-Byte-Realzahlen

```
D:\MASM>symdeb.exe
Microsoft (R) Symbolic Debug Utility  Version 4.00
Copyright (C) Microsoft Corp 1984, 1985.  All rights reserved.

Processor is [80286]
-dd 0:0
0000:0000  028B:5707 0070:0774 4A56:2C1B 0070:0774
0000:0010  0070:0774 4165:7130 F000:E831 F000:E831
0000:0020  16F6:0507 4A56:2BAD F000:E831 F000:E831
0000:0030  F000:E831 16F6:31B6 152F:0439 0070:0774
0000:0040  C000:236F F000:F84D F000:F841 0070:1F20
0000:0050  4165:0171 F000:F859 F000:E82E 4165:0150
0000:0060  F000:E8F8 0070:1A3C F000:FE6E 0070:076E
0000:0070  F000:FF53 C000:1E19 0000:0522 C000:67DB
-u4165:7130
4165:7130 FB            STI
4165:7131 50            PUSH   AX
4165:7132 53            PUSH   BX
4165:7133 51            PUSH   CX
4165:7134 52            PUSH   DX
4165:7135 57            PUSH   DI
4165:7136 56            PUSH   SI
4165:7137 1E            PUSH   DS
-
```

***Bild 1** Old-Fashioned: Symdeb zeigt den Interrupt-Vektor und den Anfang der Print-screen-Routine*

mit einem beliebigen Editor als Datei Bell.asm erstellen:

```
a
mov ah,2
mov dx,7
int 21h
mov ah,4ch
int 21h

r bx 00
r cx 11
n bell.com
w
q
```

Ahnen Sie, was das Programm macht? Wie der Name schon sagt ... – aber probieren Sie es doch selbst. Achten Sie dabei bitte auf die Leerzeile in der 7. Zeile. Der Aufruf

symdeb < bell.asm

erzeugt ein Programm Bell.com, das man starten kann – und das alles ohne Assembler und Linker. Der gleiche Effekt wird aber auch erzielt, wenn die Befehlsfolge interaktiv in Symdeb eingegeben wird. Zu beachten ist allerdings die geringfügig andere Assemblernotation: Die Angabe des Segmentpräfixes erfolgt mit cs: mov ax,[di] statt mov ax,cs:[di], und die Größe unbestimmter Operanden kann mit inc byte [0815] bzw. inc word [0815] angegeben werden.

Die Nutzung verschiedener Bildschirmseiten dient vor allem dazu, auch Programme testen zu können, die mit ihren Ausgaben den gesamten Bildschirm in Anspruch nehmen oder im Grafikmodus arbeiten. Dazu muß Symdeb mit dem Schalter /s gestartet werden. Mit dem Backslash (\) kann umgeschaltet werden. Ein weiterer nützlicher Schalter ist /k. Er erlaubt den Abbruch von Endlosschleifen mit <Ctrl>-C (^C). Das ist mit Break on im Betriebssystem nicht zu erreichen, da damit nur Systemrufe überprüft werden. Enthält die Programmschleife keine BDOS-Rufe, so wäre ein Abbruch nicht möglich. Wird allerdings gerade auf eine Eingabe gewartet, kann stets mit ^C abgebrochen werden.

Ferner ist Symdeb für die Hochsprachenunterstützung ausgelegt. So kann man beispielsweise Daten im Real-Format *dumpen* – allerdings müssen sie dem IEEE-Format entsprechen, also 4, 8 oder 10 Byte lang sein. Borlands eigenwilliges 6-Byte-Format bei Turbo-Pascal 3.0 wird also nicht bedient. Statt dessen wollen wir uns ansehen, wie man Programme ab Version 4.0 mit Symdeb symbolisch *debuggen* kann – nur als Beispiel, denn nötig hat man es angesichts der neuen Möglichkeiten eigentlich nicht mehr.

In der integrierten Entwicklungsumgebung von Turbo-Pascal ist dazu der Schalter Options/Linker/Map File auf Detailed zu setzen, beim Kommandozeilencompiler der Schalter /GD. Während der Übersetzung wird jetzt ein Map-File angelegt. Es handelt sich um ein Textfile, das die Zuordnung der Bezeichner zu den Adressen enthält und das man sich mit einem Editor ansehen kann. Das Programm Mapsym wandelt das Map-File in ein Sym-File um, wie es von Symdeb verwendet wird:

mapsym test

Dieses kann folgendermaßen in den Debugger geladen werden:

symdeb test.sym test.exe

Der Debugger zeigt beim Kommando u (Unassemble) Pascal- und Maschinenbefehle gemischt an. An Stelle der Adressen können die Variablen- bzw. Prozedurnamen angegeben werden.

Moderne Nutzeroberflächen machen vor Debuggern nicht halt. Microsoft stellte mit Codeview den ersten Fullscreen-Debugger vor – für Symdeb-Umsteiger befehlskompatibel. So ist es möglich, den Quellcode, die Registerbelegung und mehrere Speicherbereiche als Dump gleichzeitig in verschiedenen Fenstern zu beobachten (siehe Bild 2). Auch die Maus kann benutzt werden. Allerdings wird der erhöhte Komfort auch mit größerem Einarbeitungsaufwand bezahlt, so daß Gelegenheits-Bugjäger ihn wohl kaum benutzen werden. Tafel 2 zeigt die Funktionstastenbelegung; wie Codeview gestartet wird, läßt sich bei Hübener, J.: MS-DOS, Verlag Technik, Berlin 1989, S. 183 nachlesen.

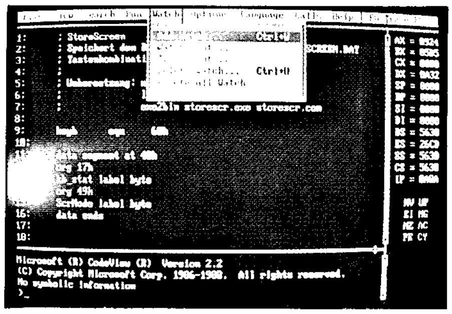


Bild 2 Microsofts Debugger-Flaggschiff Codeview

Tafel 2 Funktionstastenbelegung bei Codeview

F1	Öffnen des Hilfefensters
F2	Öffnen des Registerfensters
F3	Umschaltung zwischen Quelltext und Assemblercode
F4	Umschalten auf die Bildschirmausgaben des Programms
F5	Starten des Programms (Go)
F6	Aktuelles Fenster wechseln
F7	Starten des Programms und Abarbeiten bis zur Cursorposition
F8	Schrittbetrieb (Trace)
F9	Unterbrechungspunkt an der Cursorposition setzen
F10	Schrittbetrieb ganzer Funktionen (entspricht PTrace bei Symdeb)

Codeview unterstützt auch die Microsoft-Hochsprachen (Fortran, Basic und C; neuere Versionen auch Pascal). Hier wird aber ein anderer Weg gegangen als bei Borland, wo sich der Debugger stets ein gleichnamiges Quellfile sucht. Durch Setzen des Schalters /zi beim Compilieren bzw. Assemblieren werden Zeilennummern und Quellcode in das Objektfile eingefügt, beim Linken ist Schalter /co zu verwenden. Das erzeugte Exe-File enthält jetzt auch den Quelltext und Codeview benötigt nur das Exe-File, um auf Hochsprachenniveau zu arbeiten. Das fehlerfreie Programm ist dann noch einmal ohne den entsprechenden Schalter zu linken, damit der Quellcode nicht zugänglich ist.

Voraussetzung für Quellcodeunterstützung ist ein Linker ab Version 3.6 aus dem MASM 5.0-Paket. Sollen so vorbereitete Programme mit dem Turbo-Debugger bearbeitet werden, so muß das dort mitgelieferte Utility Tdconvrt, verwendet werden.

Die neuen Quick-Compiler von Microsoft warten inzwischen auch mit integrierten Debuggern auf.

Neben Codeview existieren noch eine ganze Reihe anderer Fullscreen-Debugger, die bestimmte Vorzüge besitzen und sich größtenteils auf das Microsoft-Sym-Format stützen. Sehr nutzerfreundlich präsentiert sich AFD (Advanced Fullscreen Debugger). Ähnlich wie bei Codeview werden das Programm, die Register und zwei Dump-Bereiche gleichzeitig dargestellt (Bild 3). Insbesondere das Abspeichern von Breakpoints und die Nutzung von Schleifenzählern zum Abbruch nach einer bestimmten Anzahl von Durchläufen erleichtern die Arbeit (Bild 4). Eine weitere Hilfe ist das Speichern von Befehlsfolgen zur wiederholten Abarbeitung. Das kann entweder durch Erstellen der Befehle mit einem Editor erfolgen (Befehlsübersicht siehe Tafel 3), oder durch Einschalten des Teach-in-Modus, der alle abgearbeiteten Kommandos speichert. Beispielsweise wird die Kommandofolge

```
KX1 testafd
m 1 ds:100
r cx=0815
```

in der Datei Auto.dat automatisch abgearbeitet, wenn AFD mit

afd "XX auto.dat

gestartet wird. Sie bewirkt, daß das Pro-

```
AX 0815   SI 0000   CS 5A51   IP 0100   Stack +0 0000          FLAGS 0200
BX 4711   DI 0000   DS 5A51                   +2 0000
CX 0000   BP 0000   ES 5A51   HS 5A51         +4 0000   OF DF IF SF ZF AF PF CF
DX 0000   SP FEEE   SS 5A51   FS 5A51         +6 0000    0  0  1  0  0  0  0  0
CMD >                                               1          0  1  2  3  4  5  6  7
                                               --00-  DS:0000    70 69 74 A0 00 9A EE FE
                                                      DS:0008    1D F0 ED 04 20 20 20 01
0109 204861        AND     [BX+SI+61],CL              DS:0010    61 6C 6C 65 73 20 20 20
010C 7264          JC      0172                       DS:0018    72 6F 67 65 72 FF FF FF
010E 204469        AND     [SI+69],AL                 DS:0020    FF FF FF FF FF FF FF FF
0111 736B          JNC     017E                       DS:0028    FF FF FF FF 4A 5A E4 FF
0113 2025          AND     [DI],AH                    DS:0030    46 4A 14 00 18 00 51 5A
0115 64            DB      64                         DS:0038    FF FF FF FF 00 00 00 00
0116 00666F        ADD     [BP+6F],AH                 DS:0040    00 00 00 00 00 00 00 00
0119 7220          JC      013B                       DS:0048    00 00 00 00 00 00 00 00

2          0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
DS:FFD0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........  ........
DS:FFE0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........  ........
DS:FFF0    00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........  ........
DS:0000    70 69 74 A0 00 9A EE FE  1D F0 ED 04 20 20 20 01  pit.....  ....   .
DS:0010    61 6C 6C 65 73 20 20 20  72 6F 67 65 72 FF FF FF  alles     roger...

1  Step  2StepProc 3Retrieve 4  Help  5Set BRK  6     7 up  8 dn  9 le  0 ri
```

Bild 3 Der Advanced Fullscreen Debugger (AFD)

```
        B R E A K P O I N T   E N T R Y   M E N U

BR#  Break ADR  Condition                            Count  Occur  Action
  1  5A51:0190  AX=0................................     5      0  STOP
  2  5A51:0000  ....................................     0      0
  3  5A51:0000  ....................................     0      0
  4  5A51:0000  ....................................     0      0
  5  5A51:0000  ....................................     0      0
  6  5A51:0000  ....................................     0      0
  7  5A51:0000  ....................................     0      0
  8  5A51:0000  ....................................     0      0

 Disassembler window address >

     5A51:0136 7220         JC      0158
          0138 44           INC     SP
          0139 4F           DEC     DI
          013A 53           PUSH    BX
          013B 205061       AND     [BX+SI+61],DL
          013E 7274         JC      01B4
          0140 6974696F6E   IMUL    SI,[SI+69],6E6F
          0145 732E         JNC     0175
          0147 2E           CS:

1View Trace 3Read Setup 4   Help   5Main Menu  7Save Setup 8Disas.Window 9Clear
```

Bild 4 Die Breakpoint-Unterstützung von AFD

Tafel 3 Die AFD-Kommandos im Überblick

Kommando	Funktion
L datei [param] [adr]	Laden eines Files in den Speicher
W datei,adr,laenge	Schreiben einer Datei auf Diskette
R reg=wert	Register setzen
D adr	Befehlscode anzeigen
M fenster adr	Speicherbereich anzeigen
G [startadr] [breakadr]	Programmausfühung
QUIT [R[ESIDENT]]	AFD beenden
A [adr]	Assemblieren
P adr,string	Zeichenkette in den Speicher schreiben (patch)
F adr,anzahl,string	Speicherbereich füllen
S [[adr,]string]	String suchen
C adr,adr,laenge	Speicherbereiche vergleichen
CO adr,adr,laenge	Speicherbereich kopieren
I adr	Eingabe vom Port
O adr,wert	Ausgabe auf Port
BW datei	Breakpoints in Datei schreiben
BL datei	Breakpoints von Datei lesen
XT	Starten des Teach-in-Modus
XX [datei]	Ausführen einer Kommandodatei
XW datei	Kommandos in Datei speichern
XL datei	Kommandos aus Datei laden
PH adr,laenge[,datei]	Drucken Hex oder ASCII auf Drucker (Standard) oder Datei Drucken
PD adr,laenge[,datei]	von reassembliertem Code

gramm Testafd geladen, in Dump-Fenster 1 die Adresse ds:100 eingestellt und das Register cx mit dem Wert 0815 geladen wird. Die Buchstaben KX am Anfang sind eine Kennung.

Bei neueren Versionen ist sogar die Einbindung eigener Befehle möglich /4/. Beim Start läßt sich ein Interrupt festlegen, der angesprungen wird, wenn AFD einen unbekannten Befehl erkennt, beispielsweise

afd /U60

für Interrupt 60H. Ein dort installiertes eigenes Programm kann die neuen Befehle bearbeiten; ein Zeiger auf den Befehl wird in den Registern DS:SI übergeben. Die Interrupts 60H bis 65H sind für derartige Anwendungen von DOS freigehalten.

Ein anderer Fullscreen-Debugger ist Periscope. Er unterstützt als einer der ganz wenigen (neben dem noch zu besprechenden Turbo-Debugger) auch die Register des Arithmetik-Koprozessors und läßt sich ebenfalls gut zum Testen von Turbo-Programmen verwenden.

Eine ganz ausgeklügelte Hilfe ist die Verwendung zusätzlicher Hardware durch einige Debugger. Dabei befindet er sich auf einer zusätzlichen Karte mit eigenem Prozessor und eigenem RAM. Somit können ohne Absturz auch diffizilste Betriebssystemsroutinen getestet werden, da das Debugger-Programm nicht auf das Betriebssystem angewiesen ist.

Source-Level-Debugging

Es ist leicht vorstellbar, daß das beschriebene Verfahren der Programmentwicklung auf Maschinenebene moderne Hochsprachen mit komplexen Daten- und Programmstrukturen sowie teilweise recht großen Bibliotheken nur unzureichend unterstützt. Der Programmierer benötigt Informationen über die interne Darstellung der Daten, Parametervermittlung und Bibliotheksaufbau. Deshalb liegt der Gedanke nahe, direkt auf Ebene des Quellcodes den Programmverlauf und die Variablenbelegung zu beobachten. Weg-

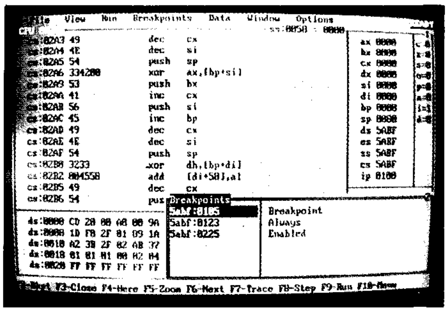


Bild 5 Das Doppelleben des Turbo-Debuggers: Auf Maschinenebene ...

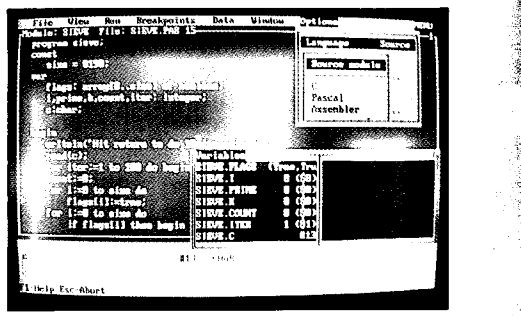


Bild 6 ... und im Hochsprachenlook

bereiter war Turbo-Pascal 5.0 mit dem integrierten Debugger in der Entwicklungsumgebung und separatem Turbo-Debugger, der noch weitergehende Möglichkeiten bietet.

Der integrierte Debugger bietet die Möglichkeit der Arbeit im Einzelschrittbetrieb (Hotkey F7). Dabei wird aber nicht ein Maschinenbefehl, sondern jeweils eine Pascal-Anweisung ausgeführt, genaugenommen eine Quelltextzeile. Um Anweisungen einzeln auszuführen, müssen sie auch einzeln auf einer Zeile stehen. Bereits ausgetestete Prozeduren lassen sich auch in einem Schritt ausführen (Hotkey F8). Bibliotheksfunktionen werden stets im Ganzen abgearbeitet, da sie keine Debug-Informationen enthalten. Eingegrenzten Fehlern kann man sich gut mit *go to cursor* (Hotkey F4) nähern. An der Cursorposition wird das Programm gestoppt, um in Einzelschritten weiterzuarbeiten. Eventuelle Ausgaben des Testprogrammes erfolgen – wie bei Symdeb möglich – generell in eine separate Bildschirmseite, zu der umgeschaltet werden kann.

Wird mit Includes oder Units gearbeitet, so lädt Turbo automatisch die Quelltexte der anderen Programmteile nach, wenn Prozeduren oder Funktionen daraus aufgerufen werden. Erfolgt während der Trace-Schritte eine Veränderung des Quelltextes, so erkennt der Compiler dies und fragt, ob neu übersetzt werden soll. Die Möglichkeit der sofortigen Compilierung nach einer Fehlerkorrektur ist auch der entscheidende Vorteil gegenüber dem Stand-Alone-Debugger. Die Menüs *Debug* und *Break/Watch* bieten noch folgende Möglichkeiten (siehe auch MP 11/89, S. 325): Setzen und Löschen von Unterbrechungspunkten (Breakpoints), Beobachtung von Variablenbelegungen (Watch), Berechnung und Änderung von Pascal-Ausdrücken (Evaluate) sowie Informationen über die Prozedurhierarchie (Call Stack).

Nicht unterstützt werden dagegen bedingte Unterbrechungspunkte; das Verfolgen von Zeigern ist unkomfortabel, da sie nur als Adressen dargestellt werden.

Sie werden sich fragen, warum neben dem integrierten Debugger noch ein Stand-Alone-Debugger ausgeliefert wird. Aus Speicherplatzgründen konnten nur die wichtigsten und am häufigsten benötigten Funktionen aufgenommen werden. Für weitergehende Aufgaben muß der Turbo-Debugger (der auch für Turbo-C geeignet ist) herangezogen werden. Außerdem unterstützt er den Turbo-Assembler und erlaubt gleichzeitig das Debugging auf Maschinenebene. Er bietet dem Programmierer seine gewohnte Arbeitsumgebung, dazu gehören die bewährten Menüs, kontextbezogene Hilfe und die Zahlendarstellung im Format der jeweiligen Sprache ($4711 bei Pascalprogrammen bzw. 0×4711 in C). Ein Fenster mit den Maschinenbefehlen, den Registern und dem Dump oder mit dem Quelltext ist stets präsent, weitere können vom Nutzer beliebig geöffnet und angeordnet werden (weitere Dumps, Variablenbelegungen, Breakpoints, Koprozessor-Register und vieles mehr). Die Bilder 5 und 6 geben einen kleinen Einblick.

Um ein Programm mit Informationen für das Stand-Alone-Debugging zu versorgen, muß in der integrierten Entwicklungsumgebung die Option *Debug/Stand Alone* auf *On* gesetzt werden bzw. der Schalter /v des Kommandozeilencompilers TPC angegeben werden, also

TPC test /v

Die Programme werden dadurch erheblich größer (15 KByte statt 9 KByte bei einem Testprogramm). Nach erfolgreichem Test können diese Informationen aber wieder entfernt werden, da sie sich am Ende des Files befinden. Um nicht neu übersetzen zu müssen, stellt Borland das Programm Tdstrip zur Verfügung.

Der Turbo-Debugger nutzt nicht das Quasi-Standard-Format von Microsoft. Um trotzdem solche Programme debuggen zu können, bietet Borland das Programm Tdmap an, mit dem von Standard-Linkern erzeugte Map-Dateien (siehe oben) im Turbo-Format an das Exe-File angehängt werden. An dieser Stelle sei auch erwähnt, daß der Turbo-Linker mit dem Schalter /t direkt Com-Files erzeugt und damit Exe2bin überflüssig macht.

Der Turbo-Debugger bietet alles, was dem integrierten Debugger noch fehlt, beliebige bedingte und unbedingte Breakpoints, Abbruch auf Tastendruck sowie Unterstützung des Arithmetik-Koprozessors. Daneben gibt es zwei Optionen zur Unterstützung, sich wiederholender Einstellungen, wie sie bei mehreren Testumläufen auftreten: History Lists und Makros. Ähnlich den bereits bekannten Pick-Listen beim Laden von Files merkt sich der Debugger die zuletzt vorgenommenen Eingaben in den einzelnen Menüs. Sie können mit den Cursortasten wiederholt ausgewählt werden. Wenn mit Tdinst nichts anderes festgelegt ist, so werden die letzten zehn Eingaben festgehalten. Makros enthalten wiederholte Befehlsfolgen, aber auch Einstellungen von Bildschirmfenstern, Setzen von Breakpoints u. ä. Beliebige Makros können auf beliebige Tasten gelegt werden. Das Anlegen, Löschen und Starten von Macros erfolgt im Menü Options/Macros. Um ein neues Makro zu definieren, wird in einen Lernmodus (Teach in) geschaltet, der alle Nutzeraktionen speichert. Auf Tastendruck sind sie jederzeit wieder aufrufbar.

Der Turbo-Debugger präsentiert sich auf der Höhe der Zeit. Er unterstützt den i80386, in-

dem er Programmen eine virtuelle 8086-Task zur Verfügung stellt. Man muß dazu TD386 starten. Etwas kleiner geht es auch über die serielle Schnittstelle. So kann der Debugger auf einem zweiten Rechner laufen und ermöglicht auch das Testen großer Programme. Nötig sind die beiden Programme Tdremote und Tdrf. Es können Übertragungsraten bis zu 115 000 Baud eingestellt werden. Weiterhin kann er mit EMS umgehen bzw. dessen Emulation auf einem 286er nutzen.
Die Version 1.5 des Turbo-Debuggers unterstützt jetzt auch die objektorientierten Möglichkeiten des Compilers Turbo-Pascal 5.5 (siehe MP 2/90 S. 44).

Literatur

/1/ IBM: Disk Operating System, Technical Reference 1984
/2/ Heimsoeth Software GmbH & Co Produktions- und Vertriebs-KG: Handbuch Turbo-Pascal, Version 5.0, München, 1988
/3/ Borland International: Turbo Debugger, User's Guide Version 1.0, Scotts Valley 1988
/4/ Bäumler, D.: Die AFD-Pro-Werkstatt, Assembler-Programmierung von Anfang an. DOS Extra (1989) 8
/5/ Strauss, P.: Moderner PC-Assembler: MASM 5.0 zeigt, was er kann. mc (1989) 5, Franzis Verlag, München

Execlook

registriert Programmaufrufe

Dr. Christian Hanisch, Berlin

Bei PC-Installationen mit Laufkundschaft *ist es für den Systemverantwortlichen von Interesse zu erfahren, welche Komponenten der bereitgestellten Software mit welcher Häufigkeit zu welcher Zeit genutzt werden, um Optimierungen oder anderes an der Installation vornehmen zu können. Das hier vorgestellte Programm kann als Werkzeug u. a. für diesen Zweck herangezogen werden.*

Das Interesse der PC-Nutzer an der sogenannten *Performance* ihres Systems nimmt nach einer gewissen Zeit – wenn die Festplatte mit Programmpaketen, TSR-Programmen und Utilities langsam voll wird – eine Richtung hin zum sinnvollen Reduzieren auf das unbedingt Notwendige. Man kann das Problem dem Gefühl nach angehen und dann subjektiv und administrativ den Softwarebestand reduzieren. Man kann aber auch analytisch-statistisch vorgehen und erst einmal die Frequentierung gewisser Programme über einen längeren Zeitraum beobachten und danach dann objektiv begründet den Bestand an Standardsoftware festlegen. Letzteres kann gegebenenfalls ein permanenter Prozeß werden, der dem Systemverantwortlichen einen ständigen Überblick über Nutzung und Nutzungsgewohnheiten ermöglicht.
Eine Lösung dieses Problemkreises liefert das residente (TSR-) Programm **Execlook**, das sich nach seinem Aufruf, zum Beispiel aus der Autoexec.bat heraus, resident macht und jeden EXEC-Aufruf via INT-21H (Funktion 4BH) abfängt. Die Dateibezeichnung des auszuführenden Programms kann dann mit vollständiger Pfadangabe sowie dem Datum – in sortierfähiger Form als JJJJ.MM.TT – und der Uhrzeit hh:mm:ss als zusätzliche Information in eine Datei geschrieben werden. Das Dateiformat entspricht dem sogenannten ASCII-SDF-Format, welches auf einfache Weise gestattet, die Daten mit

APPEND FROM <Dateibezeichnung> **DELIMITED**

in eine dBase-Datenbank aufzunehmen.
Für jeden Programmaufruf über INT-21H wird ein Satz folgenden Aufbaus geschrieben:

<Dateibezeichnung>,JJJJ.MM.TT, hh:mm:ss<CRLF>

```
              E X E C L O O K    ( INT 21h )
Aufruf:
EXECLOOK [<Dateibezeichnung>][/]<CR>

<Dateibezeichnung>::=Protokolldatei
(C:\EXECLOOK.DAT)     zur Registrierung
                      aller mit der EXEC-
                      Fkt. des INT 21h
                      aufgerufenen Pro-
                      gramme. Datei muß
                      schon existieren !
/       ::= Zwangsweise neu installieren.
-----------------------------------------
if exist C:\EXECLOOK.DAT goto LOOKexist
echo.>C:\EXECLOOK.DAT
:LOOKexist
execlook C:\EXECLOOK.DAT >nul
if errorlevel 8 goto Dateifehler
if errorlevel 4 goto SchonDa
                    (C) Ch. Hanisch
```

Bild 2: Installationsmeldung und Anwendungsbeschreibung von EXECLOOK.COM

```
        E X E C L O O K
           on INT 21
       already resident !!

  Protokollieren von Programm-
           aufrufen:

  execlook [<Dateibez.>][/]<CR>

[ Test 00010000B on 0000:01FC ]
                    (C) Ch. Hanisch
```

Bild 3: Reaktion bei wiederholtem Aufruf von EXECLOOK

```
/~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\
:: EXECL.BAT  - Testdatei zum Programm EXECLOOK.COM
@echo off
                                                     echo off
if exist C:\EXECLOOK.DAT goto LOOKexist
echo P r o t o k o l l, - L i s t e,:  >C:\EXECLOOK.DAT
echo ===== 50 ========,==== 10 ====,= >>C:\EXECLOOK.DAT
:LOOKexist
execlook C:\EXECLOOK.DAT > nul
if      errorlevel 8 goto Dateifehler
if      errorlevel 4 goto SchonDa
if not errorlevel 4 goto ende
:Dateifehler
echo  Error:  Dateifehler  - C:\EXECLOOK.DAT missing,
echo                         oder ParamStr falsch o.a.
goto ende
:SchonDa
echo  Warning: EXECLOOK ist bereits installiert
echo           bzw. Flag auf 0000:1FCh gesetzt !
:ende
echo.
\________________________________________________________________/
```

Bild 1: BATCH-Datei EXECL.BAT zum Initialisierungs-Aufruf von EXECLOOK

Dabei ist die <Dateibezeichnung> maximal 60 Zeichen lang, so daß jeder Satz einschließlich des abschließenden <CRLF> maximal 80 Zeichen lang sein kann, womit eine Auswertung via Bildschirm ohne Zeilenumbruch möglich ist, wenn man in dBase die Struktur wie folgt deklariert:

01 PGMNAME,C,50 (statt 60)
02 DATUM,C,10
03 ZEIT,C,8

Execlook erwartet die Dateibezeichnung der Protokolldatei als Parameterinformation (ParamStr) beim Aufruf. Bei fehlendem Parameter wird die Datei: C:\Execlook.dat. angenommen. Absichtlich wurde das Anlegen der Protokolldatei nicht in Execlook integriert. Execlook hängt immer nur Sätze an die bereits vorhandene Protokolldatei an. Dadurch kann der Systemverantwortliche beim formellen Anlegen der Protokolldatei mit einem beliebigen Texteditor oder über Echo-Umleitungen in einer Batch-Datei (siehe Bild 1) Zusatz- und Kenninformationen vorgeben.
Der Aufruf von Execlook kann zum Beispiel mit der Batch-Datei (Execl.bat aus Bild 2) erfolgen.
Execlook schützt sich selbst gegen mehrmaligen Aufruf und damit gegen mehrfaches Residentmachen durch Setzen eines Bit-Schalters im freien User-Bereich des Betriebssystems auf der Adresse 0000:01FCH im Bit 4. Falls andere Programme an dieser Stelle das Bit 4 gesetzt haben, kann man eine zwangsweise Installation von Execlook erreichen, indem man beim Aufruf als Parameterangabe zusätzlich zur wahlweisen Angabe der Dateibezeichnung der Protokolldatei einen Schrägstrich / angibt:

EXECLOOK [<Dateibezeichnung>][/]

Execlook gibt die in Bild 2 gezeigte Installationsmeldung aus. Falls das Programm erneut aufgerufen wird, erscheint die in Bild 3 gezeigte Meldung. In BATCH-Dateien kann man die Bilder unterdrücken, indem man eine Ausgabeumleitung zum Gerät NUL (>nul) vornimmt.
Der Quelltext des Programms Execlook.asm aus Bild 4 kann mit MASM, LINK und EXE2BIN in eine abarbeitbare COM-Datei überführt werden.
Execlook hat den typischen Aufbau eines TSR-Programms aus residentem Laufzeitmodul (von Marke MY_INT21 bis TSR_END) und dem Installationsteil (von Marke INIT_VECTORS bis zum Programmende). Bei der Installation wird die ParamStr-Information aus der Aufrufzeile ausgewertet und dann der Laufzeitmodul resident gemacht. Bei jedem INT-21H-Ruf wird auf die EXEC-Funktion 4BH geprüft und dann entweder ein entsprechender Satz in die Protokolldatei geschrieben und zum originalen INT-21H verzweigt oder sofort mit einem JMPOLD an der Marke Do_It verzweigt.

```
/~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  1:     page 88,132
  2:     title EXECLOOK.ASM   -   Registrierung von EXEC-Aufrufen
  3: COMMENT @
  4:   Das Programm faengt ueber einen "neuen" INT 21 -Handler
  5:   alle via INT 21h Funktion 4Bh (EXEC-Funktion) ausgesandten
  6:   Aufrufe von Programmen ab und registriert sie in einer Datei.
  7:   Das Programm EXECLOOK sollte in einer BATCH-Datei folgender-
  8:   maßen aufgerufen werden (z.B. in der AUTOEXEC.BAT):
  9:
 10:           if exist C:\EXECLOOK.DAT goto LOOKexist
 11:           echo Protokoll-Datei, der PGM-,Rufe:>C:\EXECLOOK.DAT
 12:           :LOOKexist
 13:           execlook C:\EXECLOOK.DAT >nul
 14:           if errorlevel 8 goto Dateifehler
 15:           if errorlevel 4 goto SchonDa
 16:           ...          ...
 17:
 18:   Der Name der Protokolldatei ist beliebig. Diese Datei muß
 19:   aber existieren, EXECLOOK fügt ans Ende der Datei die ent-
 20:   sprechenden Einträge nur noch an:
 21:
 22:   <Dateibezeichnung>,<Datum als JJJJ.MM.TT>,<Uhrzeit><CRLF>
 23:        ...                ...
 24:   <Dateibezeichnung>,<Datum als JJJJ.MM.TT>,<Uhrzeit><CRLF>
 25:
 26:   Damit ist eine Auswertung nach Überführung in eine
 27:   dBASE-Datenbank folgender Struktur möglich:
 28:           ------------------------------------
 29:           01 PGMNAME,C,50    --->  X ... ... X
 30:           02 DATUM,C,10      --->  JJJJ.MM.TT
 31:           03 ZEIT,C,8        --->  hh:mm:ss
 32:           ------------------------------------
 33:    Die Überführung geschieht mit:
 34:           ------------------------------------
 35:           USE <name_DBD>
 36:           APPEND FROM C:\EXECLOOK.DAT DELIMITED
 37:           ------------------------------------
 38:                  (C)  Ch. Hanisch
 39: @                                                      22.07.89
 40:
 41: ;    M a c r o s:
 42: DOS   macro func,parm
 43:       IFNB <parm>
 44:       mov  ax,func*256+parm
 45:       ELSE
 46:       mov  ah,func
 47:       ENDIF
 48:       int  21h
 49:       endm
 50:
 51: LINK  macro int,oldintloc,newintloc
 52:       mov al,int
 53:       DOS 35h
 54:       mov Word PTR oldintloc,bx
 55:       mov Word PTR oldintloc+2,es
 56:       mov dx,OFFSET newintloc
 57:       mov al,int
 58:       DOS 25h
 59:       endm
 60:
 61: KEEP  macro initbegin,startpgm
 62:       mov  ax,3100h            ; T)erminate and S)tay  R)esident
 63:       mov  dx,((OFFSET initbegin-OFFSET startpgm)+10Fh) SHR 4
 64:       int  21h
 65:       endm
 66:
 67: JMPOLD macro isr
 68:         jmp DWord PTR isr
 69:         endm
 70:
 71: PUSHREGS macro
 72:        push ax
 73:        push bx
 74:        push cx
 75:        push dx
 76:        push bp
 77:        push si
 78:        push di
 79:        push ds
 80:        push es
 81:           endm
 82:
 83: POPREGS macro
 84:        pop  es
 85:        pop  ds
 86:        pop  di
 87:        pop  si
 88:        pop  bp
 89:        pop  dx
 90:        pop  cx
 91:        pop  bx
 92:        pop  ax
 93:           endm
 94:
 95: ; ---------------------------------------------------------------------
 96:
 97: code      SEGMENT  'code'
 98:           ASSUME cs:code,ds:nothing,es:nothing
 99:           org 100h
100: TSR_Begin: jmp INIT_VECTORS ; Initialize and Attach to DOS
101: Kanal   dw   ?               ; Handle-Nummer
102: PGMName db   80 DUP (?)      ; Zwischenspeicher für Protokoll
103: ;     =========>  " <PGMName>,<JJJJ.MM.TT>,<hh:mm:ss><CRLF>"
104: dsn       db  'C:\'          ; Laufwerk C:\  - ROOT-Directory
105:           db  'EXECLOOK'     ; <filename>
106:           db  '.DAT'         ; .<type>
107:           db  17 DUP (00h)
108: OLD_INTLOC label DWORD       ; Address for old INT 21-Routine
109:           dd ?
110: ; =====================================================================
111: MY_INT21 PROC
112:           pushf
113:           cmp  ax,4B00h      ; EXEC-Funktion ?
114:           je   LookIt
115:           jmp  Do_It
116: ; ---------------------------------------------------------------------
117: Anzahl    dw    0               ; CX ::= Schreibe Anzahl Bytes
118: Datum     db    ',jjjj.mm.tt'   ; Datum (sortierfähig) und Uhrzeit
119: Zeit      db    ',hh:mm:ss'     ; SDF-Datei: Komma-begrenzt.
120:           db    13,10           ; 22 Zeichen anfügen!
121: ; ---------------------------------------------------------------------
122: LookIt:  sti
123:          PUSHREGS
124: ;        Registrierung der aufzurufenden Programme:
125: ;                        ES:BX  ::= Parameterblock
126: ;                        DS:DX  ::= ASCIIZ-Dateispezifikation
127:          push cs
128:          pop  es              ; ES identisch CS setzen
129:          mov  si,dx
130:          xor  di,di
131: Cyclus:  mov  al,ds:[si]
132:          or   al,al
133:          jz   AmEnde
134:          cmp  di,58           ; Max. Laenge von PGMName
135:          jae  AmEnde
136:          mov  Byte PTR cs:PGMName[di],al
137:          inc  si
138:          inc  di
139:          jmp  Short Cyclus
140: AmEnde:  push di
141:          push cs
142:          pop  ds              ; DS identisch CS setzen
143: ; DATUM und UHRZEIT ermitteln:
144: ; ============================
145: ; Aktuelles Datum:
146:          mov  ah,2Ah ; Datum abfragen - Binaer in:
147:          int  21h    ; AL := Wochentag (0=Sonntag .. 6=Samstag)
148:                      ; CS := Jahr  (1980 .. 2099)
149:                      ; DH := Monat (1 .. 12)
150:                      ; DL := Tag   (1 .. 31)
151: ; Konvertieren in Zeichenkette JJJJ.MM.TT (Datum sortierfähig)
152:          push dx
153:          mov  ax,cx
154:          mov  cx,3004h
155:          mov  di,OFFSET CS:Datum+5  ; Jahr
156:          call FORM_16
157:          pop  ax                       ; Monat und Tag aus DX
158:          push ax
159:          xchg al,ah
160:          xor  ah,ah
161:          mov  cx,3002h
162:          mov  di,OFFSET CS:Datum+8  ; Monat
163:          call FORM_16
164:          pop  ax
165:          xor  ah,ah
166:          mov  cx,3002h
167:          mov  di,OFFSET CS:Datum+11 ; Tag
168:          call FORM_16
169: ; Aktuelle Uhrzeit:
170:          mov  ah,2Ch   ; Uhrzeit abfragen - Binaer in:
171:          int  21h      ; CH := Stunden  (0 .. 23)
172:                        ; CL := Minuten  (0 .. 59)
173:                        ; DH := Sekunden (0 .. 59)
174: ; Konvertieren in Zeichenkette hh:mm:ss
175:          mov  al,ch                 ; Stunden  nach AL
176:          mov  ch,dh                 ; Sekunden nach CH
177:          xor  ah,ah
178:          push cx
179:          mov  cx,3002h
180:          mov  di,OFFSET CS:Zeit+3   ; Stunden
181:          call FORM_16
182:          pop  ax
183:          push ax
184:          xor  ah,ah
185:          mov  cx,3002h
186:          mov  di,OFFSET CS:Zeit+6   ; Minuten
187:          call FORM_16
188:          pop  ax
189:          xchg al,ah
190:          xor  ah,ah
191:          mov  cx,3002h
192:          mov  di,OFFSET CS:Zeit+9   ; Sekunden
193:          call FORM_16
194: ;
195:          mov  si,OFFSET ds:Datum    ; Quelle ist DS:[si]
196:          mov  ax,OFFSET ds:PGMName  ; Ziel   ist ES:[di]
197:          pop  di
198:          add  ax,di
199:          add  di,22         ; Max. ===> 58 + 22 = 80
200:          mov  Anzahl,di     ; Anzahl auszugebender Zeichen
201:          mov  di,ax
202:          mov  cx,22         ; Laenge+2 (<CRLF>)
203:          REP  movsb
204: ; OPEN Handle:                 ( DS ist identisch CS )
205:          mov  ah,3Dh        ; OPEN Kanal
206:          mov  al,01h        ; Modus: OPEN für Schreiben
207:          mov  dx,OFFSET ds:dsn
208:          int  21h
209:          jc   Fehler1       ;  AX:= Kanal-Nr. bzw. Fehler-Nr.
210:          mov  Kanal,ax      ;  Kanal-Nr. sichern
211: ; Kanalposition einstellen: move file pointer
212:          mov  ah,42h
213:          mov  al,02h        ; Relativ zu EOF
214:          mov  bx,Kanal
215:          mov  cx,0
216:          mov  dx,0          ; Verschiebung um 0 Byte
217:          int  21h
218:          jc   Fehler2       ; AX:= Fehler-Nr.
219: ; Schreiben auf Handle: Write handle
220:          mov  cx,Anzahl     ; Anzahl Bytes zu schreiben
221:          mov  ah,40h
222:          mov  bx,Kanal
223:          mov  dx,OFFSET ds:PGMName
224:          int  21h
225: ; Datei schließen: CLOSE handle
226: Fehler2:
227:          mov  ah,3Eh
228:          mov  bx,Kanal
229:          int  21h
230: Fehler1:
231:          POPREGS
232:          cli
233: Do_It:   popf
234:          JMPOLD cs:OLD_INTLOC            ;Rufe originalen INT 21h
235: ; ---------------------------------------------------------------------
```

```
236: ; 16-Bit-Zahl formatieren ins CHARACTER-Format
237: ; Adaption der Routine aus: Schild,G.
238: ; ========      "Zehn mal so schnell wie Norton"
239: ;               PC-PLUS Nr. 2 vom 11.1.89, S. 108-119
240: ; -----------------------------------------------------
241: ; "FORM_16" konvertiert eine 16-Bit-Binaerzahl
242: ; in die CHARACTER-Darstellung.
243: ; AX      := Binaerzahl (16 Bit)
244: ; CL      := Stellenzahl, d. h. Anzahl Zeichen fuer
245: ;            CHARACTER-Darstellung
246: ; CH      := Fuell-Zeichen fuer fuehrende Nullen, z. B. 20h
247: ; [DI]    := Fiktive Kommastelle im Zielstring der
248: ;            konvertierten Zahlendarstellung.
249: ;            CHARACTER-Darstellung wird nach links hin
250: ;            ab [di] entwickelt.
251: FORM_16  proc near
252:          push  ax
253:          push  bx
254:          push  dx
255:          mov   bx,10
256: f1:      xor   dx,dx       ; Zahl/10
257:          div   bx
258:          add   dl,"0"      ; Rest speichern
259: f2:      dec   di
260:          mov   [di],dl
261:          dec   cl
262:          jz    f3
263:          and   ax,ax       ; Alle Ziffern umgewandelt?
264:          jnz   f1
265:          mov   dl,ch       ; dann Fuellzeichen speichern
266:          jmp   short f2
267: f3:      pop   dx
268:          pop   bx
269:          pop   ax
270:          ret
271: FORM_16  endp
272: MY_INT21 endp
273: TSR_End  equ $

274: ; ------------------------------------------------------------
275: ;   I n i t i a l i s i e r u n g :
276: ; ------------------------------------------------------------
277:                 ASSUME ds:code,es:nothing
278: ; >>>>>>>>>>>>>>>>>> Hier geht's los !!!! <<<<<<<<<<<<<<<<<<
279: INIT_VECTORS PROC near
280:       push  cs              ; Initialisieren des Daten-Segments
281:       pop   ds
282:       xor   dx,dx           ; DX definiert setzen auf Null
283:       xor   ax,ax           ; AX definiert setzen auf Null
284:       mov   es,ax           ; ES definiert setzen auf Null
285:       mov   si,80h
286:       lodsb
287:       or    al,al           ; Ueberhaupt ParamStr-Information ?
288:       jz    Kein_ParamStr
289:       add   si,ax
290:       mov   Byte PTR ds:[si],00h ; Ende markieren !!
291:       mov   si,81h
292: Blank:
293:       lodsb
294:       or    al,al
295:       jz    Kein_ParamStr
296:       cmp   al,20h          ; <SP>  uebergehen
297:       je    Blank
298:       cmp   al,09h          ; <TAB> uebergehen
299:       je    Blank
300:       cmp   al,"/"
301:       jne   Blank
302:       and   Byte PTR es:[01FCh],11101111B
303: Kein_ParamStr:
304:       test  Byte PTR es:[01FCh],00010000B
305:       jz    HDgo            ; "Schon-Da"-Flag on 01FCh gesetzt ?
306:       jmp   SchonDa         ; Freier Nutzerbereich: 0180h..01FFh
307: HDgo:
308:       or    Byte PTR es:[01FCh],00010000B
309: ;
310: ; <ParamStr>-Information aus der Aufrufzeile:
311: ; Aufruf:
312: ; =======     EXECLOOK [<Dateibezeichnung>][/]<CR>
313:        mov   si,80h
314:        lodsb
315:        or    al,al
316:        jz    SchonFertig
317:        mov   si,81h
318:        mov   di,0
319: Leer:  lodsb
320:        or  al,al
321:        jz  Fertig
322:        cmp al,20h
323:        je  Leer
324:        cmp al,09h
325:        jz  Leer
326:        cmp al,"/"
327:        je  Leer
328:        cmp al,"a"          ; Umwandlung in GROSS-
329:        jl  HDok            ; buchstaben
330:        cmp al,"z"
331:        jg  HDok
332:        sub al,20h

333: HDok: mov Byte PTR ds:dsn[di],al
334:        inc di
335:        mov ax,di
336:        cmp ax,32 ; <Dateibezeichnung> max. 32 Zeichen lang
337:        je  Errorl
338:        jmp Short Leer
339: Fertig:
340:        or  di,di
341:        jz  SchonFertig
342:        mov Byte PTR ds:dsn[di],00h ; ASCIIZ abschließen mit: 00h
343:        inc di
344:        mov ax,di
345:        cmp ax,32
346:        je  Schonfertig
347:        jmp Short Fertig
348: SchonFertig:
349: ; OPEN Handle zum Test der Protokolldatei
350:        mov    ah,3Dh
351:        mov    al,00h          ; Versuchsweise OPEN für Lesen
352:        mov    dx,OFFSET ds:dsn
353:        int    21h
354:        jc     Error
355: ; CLOSE Handle
356:        mov    ah,3Eh
357:        int    21h
358:        jc     Error
359:        LINK   21h,OLD_INTLOC,MY_INT21
360:        mov    dx,OFFSET message   ; Installations-Meldung
361:        mov    ah,9
362:        int    21h
363: ; ------------------------------------------------------------
364:        KEEP   TSR_End,TSR_Begin
365: ; ------------------------------------------------------------
366: SchonDa:
367:      mov dx,OFFSET Errormessage ; Fehlermeldung
368:      mov ah,9
369:      int 21h
370:      mov ax,4C04h                ; errorlevel 4
371:      int 21h                     ; Return  to DOS
372: Errorl:
373:       mov  dx,OFFSET Parmfalsch
374:       mov  ah,9
375:       int  21h
376: Error: mov dx,OFFSET ParmErr
377:       mov ah,9
378:       int 21h
379:       xor ax,ax
380:       mov es,ax
381:       and Byte PTR es:[01FCh],11101111B
382:       mov ax,4C08h               ; errorlevel 8
383:       int 21h                    ; Return to DOS
384: Parmfalsch db 13,10,'ParamStr-Information fehlerhaft !!!',24h
385: ParmErr db 13,10,27,'[K',13,10,27,'[K'
386:         db 27,'[7m Error: <Dateibezeichnung> fehlerhaft'
387:         db ' ODER Laufwerk ¦ Datei existiert nicht ! '
388:         db 27,'[m',13,10,27,'[K',24h
389: ; ------------------------------------------------------------
390: message db 13,10,27,'[K',13,10,27,'[K'

391:  db             '╔══════════════════════════════════════════╗'
392:  db 13,10,'     ║          ',27,'[7m E X E C L O O K ',27,'[m'
393:  db '     ( INT 21h ) ║'
394:  db 13,10,'     ║ Aufruf:                                  ║'
395:  db 13,10,'     ║ EXECLOOK [<Dateibezeichnung>][/]<CR>     ║'
396:  db 13,10,'     ║                                          ║'
397:  db 13,10,'     ║ <Dateibezeichnung>::=Protokolldatei      ║'
398:  db 13,10,"     ║ (C:\EXECLOOK.DAT)    zur Registrierung   ║"
399:  db 13,10,"     ║                      aller mit der EXEC- ║"
400:  db 13,10,'     ║                      Fkt. des INT 21h    ║'
401:  db 13,10,'     ║                      aufgerufenen Pro-   ║'
402:  db 13,10,'     ║                      gramme. Datei muß   ║'
403:  db 13,10,'     ║                      schon existieren !  ║'
404:  db 13,10,'     ║ /      ::= Zwangsweise neu installieren. ║'
405:  db 13,10,'     ╟──────────────────────────────────────────╢'
406:  db 13,10,'     ║ if exist C:\EXECLOOK.DAT goto LOOKexist  ║'
407:  db 13,10,'     ║ echo.>C:\EXECLOOK.DAT                    ║'
408:  db 13,10,'     ║ :LOOKexist                               ║'
409:  db 13,10,'     ║ execlook C:\EXECLOOK.DAT >nul            ║'
410:  db 13,10,'     ║ if errorlevel 8 goto Dateifehler         ║'
411:  db 13,10,'     ║ if errorlevel 4 goto SchonDa             ║'
412:  db 13,10,'     ╚═════════════════ (C) Ch. Hanisch ═══════╝'
413:  db 13,10,27,'[K',24h
414: Errormessage db  13,10,27,'[K',13,10,27,'[K'
415:  db 13,10,'
416:  db 13,10,'      ╔═══════════════════════════════════╗'
417:  db 27,'[7m E X E C L O O K ',27,'[m          ║'
418:  db 13,10,'      ║             on INT 21             ║'
419:  db 13,10,'      ║         already resident !!       ║'
420:  db 13,10,'      ║                                   ║'
421:  db 13,10,'      ║    Protokollieren von Programm-   ║'
422:  db 13,10,'      ║        aufrufen  via  INT 21h     ║'
423:  db 13,10,'      ║                                   ║'
424:  db 13,10,'      ║  execlook [<Dateibez.>][/]<CR>    ║'
425:  db 13,10,'      ║                                   ║'
426:  db 13,10,'      ║ [ Test 00010000B on 0000:01FC ]   ║'
427:  db 13,10,'      ╚═══════════ (C) Ch. Hanisch ═══════╝'
428:  db 13,10,27,'[K',13,10,27,'[K',24h
429: INIT_VECTORS ENDP
430: code ENDS
431:      end TSR_Begin
```

Bild 4: Quelltext des Programms EXECLOOK.ASM

Literatur

/1/ MS-DOS Technical Reference Encyclopedia. – Microsoft Press 1986

/2/ Smode, D.:
MS-DOS für Insider. – München: Franzis Verlag, 1987

/3/ Philipps, Ch.:
Tastaturabfrage leichtgemacht. – München: Franzis-Verlag, mc (1988) 9, S. 124–131

TERMINE

II. Amiga-DeskTopVideoWerkstatt

WER? HdjT-Computerclub und Interessengemeinschaft der Computergrafik- und Videoanwender

WANN? 7. und 8. September 1990

WO? Berlin, Haus der jungen Talente

WAS?
- alles zum Gebiet Desktop-Video
- Hard- und Softwarevorführungen
- Seminare, Workshops, Präsentationen

WIE? Anmeldungen bitte an: HdjT-Computerclub, Stefan Seeboldt, Klosterstraße 68, Berlin, 1020; Tel. 2 10 32 65, Mailbox 2 12 67 45 (300/1200). Die Tagungskosten betragen 30,–.

Seeboldt

Prüfmuster für Register- und Busstrukturen

Jörg Moebes, Dr. Veronika Sauer, Bernd Däne, Lutz Finn, Ilmenau

Die Verbreitung von Mikroprozessoren in nahezu allen Lebensbereichen erfordert eine hohe Zuverlässigkeit der Systeme. Deshalb ist es notwendig, im System existierende Fehler aufzudecken und sie gegebenenfalls zu beheben oder zumindest zu berücksichtigen. Für die Fehlererkennung in Register- und Busstrukturen von 16-Bit-Mikrorechnern sollen die im folgenden Beitrag beschriebenen Prüfmusterfolgen dienen.

Geht man von komplexen digitalen Strukturen aus, die über n Eingänge und m Zustände verfügen, so lassen sich insgesamt 2^{n+m} Gesamtzustände unterscheiden, das heißt, vollständige Testmusterfolgen müßten eine Länge von 2^{n+m} haben. Bei hochintegrierten Systemen würde das „Durchspielen" aller möglichen Testmuster Testlaufzeiten ergeben, die in der Größenordnung von Jahren liegen können. Es ist also notwendig, eine Testmusterfolge zu generieren, die mit möglichst wenigen Testmustern viele Fehler erkennen läßt, um einen effektiven und sinnvollen Test durchführen zu können. Eine Hauptaufgabe der Testvorbereitung besteht demzufolge in der Erzeugung einer sinnvollen Testmusterfolge, die den oben erwähnten Qualitätsansprüchen gerecht wird. Nach /1/ lassen sich drei wesentliche Methoden bei der Testmustererzeugung unterscheiden, die

- manuelle
- pseudozufällige
- algorithmische.

Es hat sich gezeigt, daß für hochkomplexe Schaltungen, wie LSI-Bausteine (z. B. Mikroprozessoren), die Testmuster in erster Linie manuell erzeugt werden, und zwar auf der Basis geeigneter Fehlermodelle. Die Erzeugung der Testmuster muß nach Kriterien erfolgen, die für die Anwendung auf bestimmte Prüflinge abgestimmt sind. Eine Analyse physikalischer Fehlermöglichkeiten zeigt, daß es von entscheidender Bedeutung ist, Fehler an Verbindungsleitungen (Bussen) und in Registerstrukturen von Mikroprozessorsystemen zu erkennen. Typische Fehlermöglichkeiten zeigt Bild 1.

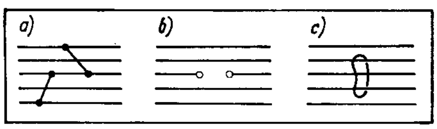


Bild 1 Typische Fehlermöglichkeiten
a) Kurzschluß von Leitungen
b) Unterbrechungen
c) gegenseitige Beeinflussung

Herleitung einer Prüfmusterfolge

Da Registerzellen und Datenbusleitungen unbedingt zur Voraussetzung von Computeraktionen gehören, muß deren einwandfreie Funktion nachgewiesen werden /2/, /3/. Dazu gehört sowohl das Ausschließen von Stuck-at-Fehlern (hervorgerufen durch Leitungsunterbrechungen, Kurzschlüsse oder fehlerhafte Flip-Flop-Strukturen bei Registern) als auch das Abtesten gegenseitiger Beeinflussungen von Leitungen oder Speicherzellen. Man spricht von Stuck-at-Fehlern, wenn Signalpegel auf den logischen Werten Null oder Eins verbleiben (Haftfehler). Es ist jedoch noch ein Unterschied zwischen Datenbusfehlern und Fehlern bei den Chipstrukturen zu machen. Während die gegenseitige Beeinflussung bei Datenbusleitungen schon durch unterschiedliche Pegel (Low oder High) feststellbar ist, muß bei den Registerzellen eine genauere Betrachtung erfolgen. Ursache dafür ist das Halbleitermaterial des Chips. Für die Fehleraufdeckung spielt auch die Polarität, das heißt, die Richtung der Pegelveränderung (Low → High oder High → Low) eine wesentliche Rolle, ein einfacher Pegelunterschied reicht im allgemeinen nicht aus. Diese Erscheinung soll im weiteren als „diodenartiger Kurzschluß" bezeichnet werden (Bild 2).

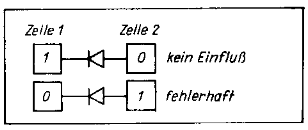


Bild 2 Zellenfehler (richtungsabhängig)

Es muß insgesamt eine Testfolge für Datenbusleitungen und Registerzellen gefunden werden, die bei ihrer Auswertung eine größtmögliche Fehleraufdeckung bei geringem Aufwand gewährleistet. Auf den ersten Blick würde eine solche Anforderung eine Prüffolge im Schachbrettmuster erfüllen (Tafel 1). Hier wird zwar auch der „diodenartige Kurzschluß" getestet, jedoch nur für einen Teil der denkbaren Fehlermöglichkeiten. Ziel des Tests soll aber die Überprüfung aller Verkopplungsvarianten sein. Darum sind diese Muster nicht ausreichend. Für die folgende Prüfmusterentwicklung sei vorerst der „diodenartige Kurzschluß" vernachlässigt.

Ausgehend von einem 16-Bit-Bus muß jede einzelne Leitung gegenüber allen anderen auf eine Beeinflussung getestet werden. Das kann durch die 16 Prüfmuster in Tafel 2 erreicht werden. Das Ziel, daß sich jede Leitung potentialmäßig mindestens einmal von jeder anderen unterscheidet, läßt sich nach /4/ auch durch eine weniger aufwendige Dualkodierung erreichen. Das macht bei der 16-Bit-Struktur eine Reduzierung auf 4 Prüfmuster aus.

Jede beliebige Spalte (jeweils eine Busleitung oder Zelle repräsentierend) unterscheidet sich beim Durchlauf aller 4 Prüfmuster – darunter ist ein Transport der Prüfmuster vom EPROM zu den Registern zu verstehen – von jeder anderen mindestens einmal. Das gleiche gilt für die entsprechenden Registerzellen (siehe Tafel 3). Beispielsweise unterscheidet sich die mit ^ bezeichnete Leitung bei dem Anliegen des 1. Musters von den acht mit 1-Potential belegten Leitungen. Bei einem Kurzschluß zwischen diesen und der bezeichneten Leitung kommt es zu Potentialveränderungen, die erkennbar bzw. auswertbar sind. Mit dem Anliegen des 2. Musters werden weitere 4 Leitungen ausgetestet, da die niederwertigen acht Bit schon geprüft sind. Es werden also mit dem Anlegen eines Prüfmusters gleich mehrere mögliche Wechselwirkungen getestet.

Mit den Prüfmustern sollen aber auch 1/0- und 0/1-Übergänge der einzelnen Register (Bitstellen) getestet werden. Es ist leicht festzustellen, daß das mit diesen Mustern nicht in jeder Bitposition erreicht wird (siehe Tafel 4). Die Schlußfolgerung daraus ist, daß man ein weiteres Muster hinzufügt. Es ist aber sofort zu erkennen, daß durch das Anfügen einer Prüfmusterzeile das Problem in der höchsten und der niedrigsten Bitposition nicht gelöst wird. Darum wird besser von der Dualkodierung ausgegangen, die die 5 Zeilen (5 Muster) und 32 Spalten in Tafel 5 enthält. Auch hier treten wieder Spalten auf, die für die Testung der 0/1- und 1/0-Übergänge nicht geeignet sind. Jedoch haben wir nun noch 22 Spalten, die einem solchen Test genügen. Von diesen verbliebenen Spalten werden jedoch nur 16 benötigt, da sie auf 16-Bit-Register und -Busleitungen angewendet werden sollen. Es ist festzustellen, daß die Zeilen nicht austauschbar sind, ohne die Testung auf 0/1- und 1/0-Übergänge zu beeinflussen, während die Reihenfolge und die Auswahl der Spalten Freiheitsgrade zuläßt. In dieser Kodierung unterscheidet sich nachweislich jede Spalte in mindestens einer Bitposition

Tafel 1 Beispiel einer Prüffolge im Schachbrettmuster

0101	0101	0101	0101	1. Prüfmuster
1010	1010	1010	1010	2. Prüfmuster

Tafel 2 Maximale Prüfmusterfolge

M1 = (1000	0000	0000	0000)
M2 = (0100	0000	0000	0000)
M2 = (0010	0000	0000	0000)
⋮			
M16 = (0000	0000	0000	0001)

Tafel 3 Minimale Prüfmusterfolge

1. Muster	0000	0000	1111	1111
2. Muster	0000	1111	0000	1111
3. Muster	0011	0011	0011	0011
4. Muster	0101 ^	0101	0101	0101

Tafel 4 Bitpositionen, die für den Test der 1/0- und 0/1-Übergänge einzelner Register ungeeignet sind

0000 0000 0011 0101 ^^ ^	0000 1111 0011 0101 ^	1111 0000 0011 0101 ^	1111 1111 0011 0101 ^ ^^	ungeeignet für Test

Tafel 5 Dualkodierung, bestehend aus 32 Spalten und 5 Zeilen

```
0000 0000 0000 0000 1111 1111 1111 1111
0000 0000 1111 1111 0000 0000 1111 1111
0000 1111 0000 1111 0000 1111 0000 1111
0011 0011 0011 0011 0011 0011 0011 0011
0101 0101 0101 0101 0101 0101 0101 0101
^^|^ |||^ |||| |||^ ^||| |||| ^||| ^|^^  für Test ungeeignet
  1  2|4  5|7| 9|11 12 |14|16|18 |20|  22 }
       3    6 8 10    13 1517  19 21      } geeignete Spalten
```

von jeder anderen. Eine optische Symmetrie in den Zeilen (gleiche Anzahl von Nullen und Einsen) muß also nicht gegeben sein.

Jetzt stellt sich die Frage nach der Aufdekkung der oben erwähnten „diodenartigen Kurzschlüsse" durch die Prüfmuster. Fest steht, daß Beeinflussungen und Kurzschlüsse zwischen Leitungen oder Registerzellen nur bei unterschiedlichen Pegeln auftreten. Damit wirkt sich ein Fehler ausschließlich bei verschiedener Belegung der betroffenen Leitungen/Zellen aus, das heißt, bei gleichem Potential wird trotz eines Kurzschlußfehlers keine Störung erkannt. Da durch die Ableitung der Prüfmuster aus der Dualkodierung der Unterschied zwischen den Spalten in wenigstens einer Stelle gesichert ist, wird auch mindestens einer der beiden Fälle (0/1 oder 1/0) getestet.

0 0 1 1	Kodierungen, die bei einem Kurzschlußfehler nicht beeinflußt werden
1 0 0 1	Kodierungen, die beeinflußt werden, aber jeweils nur 25 % der Möglichkeiten ausmachen (davon wird mindestens eine getestet)

Das bedeutet, daß mit Sicherheit eine Möglichkeit des gerichteten Kurzschlusses zwischen zwei beliebigen Registerzellen geprüft und damit ein Fehler aufgedeckt wird. Die verbleibende Unsicherheit hat sich somit auf maximal 25 % der möglichen Potentialbelegungen zweier Leitungen reduziert. Allerdings sind diese 25 % noch zu hoch gegriffen, wie anhand der Beispiele in Tafel 6 erläutert wird.

Es läßt sich anhand der Beispiele leicht nachweisen, daß Spalten, in denen jeweils ein Pegel nur einmal vertreten ist, die größten Mängel bei der Fehleraufdeckung (der restlichen 25 %) haben /5/. Folglich ist es ratsam, diesen Fakt bei der Auswahl der Kodierungen zu berücksichtigen. Es waren neben den 16 benötigten Spalten der Kodierung noch weitere 6 ungenutzt. Betrachtet man alle Möglichkeiten, so läßt sich feststellen, daß genau 6 Spalten einen einzelnen Pegel aufweisen. Damit ist die Entscheidung der Auswahl gefallen (siehe Tafel 7).

Die Prüfmuster lauten hexadezimal: 00FF, 3F03, C71C, 5965, AAAA. Diese Prüfmusterfolge soll bei allen Teilstrukturen Anwendung finden, in denen Unterbrechungs-, Übergangs- oder Kurzschlußfehler auftreten können. Somit ist es notwendig, sie bei Tests von Speicherstrukturen (Register, RAM) oder Leitungen (Verbindungen zwischen einzelnen Schaltkreisen) zu benutzen. Der fehlerfreie Prüfmusterdurchlauf bildet die Grundlage für weitere Tests. Bei vielen (wenn nicht allen) Operationen sind funktionstüchtige Register und Busleitungen eine Grundvoraussetzung. Ihre Überprüfung ist zum Beispiel Bestandteil des Kerntests in Selbsttestprogrammen von Mikroprozessorsystemen /2/.

Tafel 6 Beispiele zur Veranschaulichung der Anzahl der testbaren Fehlermöglichkeiten hinsichtlich der 1/0- und 0/1-Übergänge

Beispiel 1:

Spalten:	1	2	3	4	5	6	7	8	9	12	13	14	15	16	17	18
	Übergang 0 ⊣◁⊢ 1															
	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0
	0	1	1	1	0	0	0	0	1	0	0	0	1	1	1	1
	1	0	0	1	0	0	1	1	0	0	1	1	0	0	1	1
	0	0	1	0	0	1	0	1	0	1	0	1	0	1	0	1
getestet	x	x	x	x	x	x	x	x	x	↑	x	x	x	x	x	x
	damit sind für Spalte 12 15 von 15 Fehlermöglichkeiten der Richtung 0 ⊣◁⊢ 1 getestet															
	Übergang 1 ⊣▷⊢ 0															
	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0
	0	1	1	1	0	0	0	0	1	0	0	0	1	1	1	1
	1	0	0	1	0	0	1	1	0	0	1	1	0	0	1	1
	0	0	1	0	0	1	0	1	0	1	0	1	0	1	0	1
	x	x	x	x	x	x	x	x	x	↑	x	o	x	o	x	o
	damit sind für Spalte 12 12 von 15 Fehlermöglichkeiten der Richtung 1 ⊣▷⊢ 0 getestet															
	Insgesamt: 27 von 30															

Beispiel 2:

Spalten:	1	2	3	4	5	6	7	8	9	12	13	14	15	16	17	18
	Übergang 0 ⊣◁⊢ 1															
	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0
	0	1	1	1	0	0	0	0	1	0	0	0	1	1	1	1
	1	0	0	1	0	0	1	1	0	0	1	1	0	0	1	1
	0	0	1	0	0	1	0	1	0	1	0	1	0	1	0	1
getestet	↑	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
	damit sind für Spalte 1 15 von 15 Fehlermöglichkeiten der Richtung 0 ⊣◁⊢ 1 getestet															
	Übergang 1 ⊣▷⊢ 0															
	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0
	0	1	1	1	0	0	0	0	1	0	0	0	1	1	1	1
	1	0	0	1	0	0	1	1	0	0	1	1	0	0	1	1
	0	0	1	0	0	1	0	1	0	1	0	1	0	1	0	1
	↑	x	x	o	x	x	o	o	x	x	o	o	x	x	o	o
	damit sind für Spalte 1 8 von 15 Fehlermöglichkeiten der Richtung 1 ⊣▷⊢ 0 getestet															
	Insgesamt: 23 von 30															

Tafel 7 Ausgewählte Prüfmusterfolge

	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1
	0	1	1	1	0	0	0	0	1	1	1	0	0	0	1	1	1	1	0	0	0	1
	1	0	0	1	0	0	1	1	0	0	1	0	1	1	0	0	1	1	0	1	1	0
	0	0	1	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1	0	1	1
Spalten für Test ungeeignet	1	2			5													18			21	22
Spalten für Test geeignet			3	4		6	7	8	9	10	11	12	13	14	15	16	17		19	20		

Anwendungsbeispiel für die Prüfmuster

In /5/ wird anhand eines Selbsttestprogramms unter anderem gezeigt, wie in einem Rechner die Prüfmuster beim Test des Datenbusses und der CPU (80286) eingesetzt werden. Ausgehend von einer ungetesteten Rechnerstruktur wird auf der Grundlage des CPU-Befehlssatzes ein sogenannter Kerntest ausgeführt /2/, /6/. In diesem Rahmen werden mit den Prüfmustern der Datenbus und die Registerfunktionen der CPU getestet. Für die Sicherung der Befehlssequenz sind dabei Abstandsbytes (Kodierungen von NOP- und HALT-Befehl) zwischen den einzelnen Befehlen zweckmäßig /4/, /5/, /7/, /8/. Es ist somit sinnvoll, die NOP- und die HALT-Befehle mit Hilfe der Prüfmusterspalten zu kodieren. Das heißt konkret, daß die einzelnen Prüfmusterspalten so angeordnet werden, daß das erste Prüfmuster der Kodierung von HALT und NOP (je ein Byte) entspricht. Dieser Operationskode tritt dann als erstes Prüfmusterbyte im Zusammenhang mit einem Datentransfer auf. Bei einem Fehler besteht somit die Chance, daß die Prüfmusterbytes als sinnvolle und aussagekräftige Befehle erkannt werden, wogegen sie im fehlerlosen Fall als Daten interpretiert werden.

Kodierungen entsprechend dem Befehlssatz des 80286:

HALT 11110100 B F4 H
NOP 10010000 B 90 H

Kodierung als 1. Muster:

1111 0100 1001 0000 HALT NOP

Es ist zu erkennen, daß man für eine Prüfmusterzeile 9mal die 0 benötigt. Dieser Sachverhalt läßt sich ebenfalls mit den zur Verfügung stehenden Spalten darstellen. Allerdings muß man einen Kompromiß eingehen: Es stehen nur 8 Spalten, die den „diodenartigen Kurzschluß" optimal aufdecken, zur Verfügung. Eine weitere Spalte (1, 2 oder 5) muß hinzugenommen werden. Der dadruch eingehandelte Verlust an Fehleraufdeckung ist minimal.

Anordnung der Spalten

x x x x 3 x 4 5 x 6 7 x 8 9 10 11

Die Auswahl der mit 1 beginnenden Spalten ist nun fast beliebig

12 13 14 15 3 16 4 5 17 6 7 19 8 9 10 11

Daraus ergeben sich die fünf Prüfmuster in Tafel 8. Das erwähnte Kerntestprogramm (Ausschnitt) sieht dann folgendermaßen aus:

```
MOV AX, 0F490H   ; Transport des ersten
                 ; Prüfmusters
                 ; zum Register AX →
                 ; Test Datenbus
                 ; und Registerzellen
NOP              ; Abstandsbefehl
NOP              ; Abstandsbefehl
NOP              ; Abstandsbefehl
CMP AX, 0F490H   ; Vergleich mit 1. Prüf-
                 ; muster
NOP              ; Abstandsbefehl
NOP              ; Abstandsbefehl
NOP              ; Abstandsbefehl
JZ fort          ; Sprung zu Programm-
                 ; fortsetzung
fehl :           ; Fehlerroutine
```

Tafel 8 Prüfmusterfolge mit HALT-, NOP-Kodierung des 80286 als 1. Prüfmuster

M1	=	(	1111	0100	1001	0000	)	F490H
M2	=	(	0000	0001	0111	1111	)	017FH
M3	=	(	0001	1110	1000	0111	)	1E87H
M4	=	(	0110	0010	1010	1001	)	62A9H
M5	=	(	1010	1100	0101	1010	)	AC5AH

Ein Befehl beim 80286 kann maximal 6 Byte umfassen. Es genügen deshalb nach dem Transferbefehl weitere 3 Abstandsbefehle (NOP-Befehle), um die Befehlssequenz zu erhalten, selbst wenn der erste Befehl falsch dekodiert oder interpretiert wurde.

Ausblick

In diesem Beitrag haben wir eine Möglichkeit zur systematischen Herleitung einer Prüfmusterfolge zum Testen von Register- und Busstrukturen auf der Basis geeigneter Fehlermodelle vorgestellt. Die Aussagefähigkeit einer Prüfmusterfolge bezüglich der erkennbaren Fehlermenge wird dabei wesentlich durch das benutzte Fehlermodell bestimmt. Die entwickelte Prüfmusterfolge kann beispielsweise als Ausgangspunkt für den Test von Mikroprozessorsystemen (Selbsttest), für RAM-Tests und auch für den Test von programmierbaren Peripherieelementen (wie Timer, Interrupt-Controller, DMA-Controller etc.) genutzt werden /9/.

International ist der Trend zu beobachten, daß es heute als nahezu selbstverständlich gilt, Testprobleme bereits beim Entwurf inte-

grierter Schaltungen zu berücksichtigen (Built-in-Test); damit können die Testprozesse wesentlich effektiver gestaltet werden /10/.

Literatur

/1/ Zwiehoff, G.: Ein Selbsttestkonzept für hochintegrierte Digitalschaltungen. Diss. A. RWTH Aachen 1982
/2/ Nilsson, S. A.: Selbsttestverfahren für Mikrorechner. VDI-Berichte (1978), Nr. 328, S. 77
/3/ Jain, S. K.; Susskind, A. K.: Test strategy for microprocessors. 20th Design Automation Conference 1983, S. 703
/4/ Gärtner, A.: Optimierte Selbsttestprogramme. Diss. A, RWTH Aachen, 1984
/5/ Möbes, J.: Selbstteststrategien für 16-Bit-Mikroprozessorsysteme. Diplomarbeit TH Ilmenau, 1989
/6/ Intel: iAPX 286 Hardware Reference Manual. Santa Clara, 1983
/7/ Abraham, J. A.; Thatte, S. M.: Test generation for microprocessors. IEEE Transactions on Computers, June 1980, Vol. C-29, S. 429
/8/ Hunger, A.: Mikroprozessor-Selbsttest auf der Basis des Befehlssatzes. Elektronische Rechenanlagen (1982) 1, S. 8
/9/ Abraham, J. A.; Fuji, R.: SELF-Test for microprocessors. Int. Test Conference, Nov. 1985, S. 356
/10/ Gelsinger, D.: Design and test of the 80386. IEEE Design & Test, June 1987, S. 42

✉ KONTAKT ☏

Technische Hochschule Ilmenau, Sektion Technische und Biomedizinische Kybernetik, Wissenschaftsbereich Computertechnik, PSF 327, Am Ehrenberg, Ilmenau, 6300; Tel. 740, App. 354

Digitale Nager

Von der Mausbewegung zum elektrischen Signal

Dieter Michel

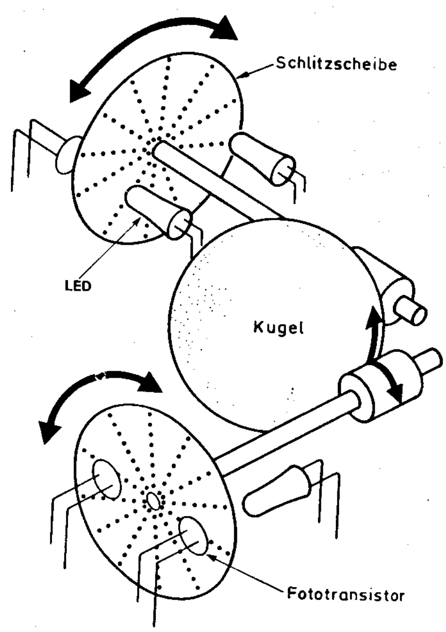
Viele Anwender orientieren sich beim Kauf einer Maus am Preis und an der mitgelieferten Software. Nach welchem Prinzip sie arbeiten, spielt dabei keine Rolle. Rollkugelmäuse unterscheiden sich von ihren optischen Verwandten aber in wesentlichen Eigenschaften – und gerade diese Unterschiede bestimmen ihr Einsatzgebiet.

Allen Mäusen gemeinsam ist das Vorhandensein von Tasten, mit denen sich eine bestimmte Funktion auslösen läßt. Obwohl bereits eine einzige Taste wie beim Apple Macintosh ausreicht, findet man im PC-Bereich meist zwei bis drei Maustasten, die mit bestimmten Funktionen belegt sind. Die Tasten sind jedoch überall gleich. Bei der Art und Weise, wie die Mäuse die Bewegungen auf der Tischplatte verarbeiten, unterscheiden sich die optischen von den mechanischen und optomechanischen Geschwistern aber erheblich.
Sogenannte optische Mäuse arbeiten ohne bewegliche Teile. Das Mausgehäuse gleitet hier über eine spezielle Oberfläche mit aufgedruckter Rasterstruktur, die von einem optischen Sensor im Innern der Maus abgetastet und in Bewegungsinformation umgesetzt wird. Zwar tasten die meisten Rollkugelmäuse die Mausbewegungen letztendlich auch optisch ab, den eigentlichen Bewegungssensoren ist jedoch eine mechanische Übertragung durch eine Rollkugel vorgeschaltet. Daher handelt es sich hier um ein optomechanisches Prinzip.
Ein wesentlicher Bestandteil des Funktionsprinzips einer optomechanischen Maus ist die bereits erwähnte Rollkugel aus Kunststoff oder Metall, die durch eine Öffnung an der Unterseite der Maus frei auf der Unterlage rollen kann. Die Kugel ist meist mit Silikon oder einem ähnlichen Belag überzogen, um die Reibung zu erhöhen und damit den Kontakt zur Unterlage zu verbessern. Sie liegt an zwei senkrecht zueinander angeordneten Laufrollen an und wird von einer gefederten dritten Rolle angedrückt. Verschiebt man das Mausgehäuse über die Tischoberfläche, so rollt die Kugel mit und setzt so die beiden Laufrollen in Bewegung. Dabei stellen die Drehachsen das Koordinatensystem dar, auf das sich die Bewegung der Maus bezieht. Um diese registrieren zu können, befindet sich auf der Welle jeder Laufrolle eine sich mitdrehende Scheibe mit radial verlaufenden Schlitzen.
Da diese Welle bei ausreichender Reibung der Kugel kraftschlüssig mit der Tischoberfläche gekoppelt ist, hängt der Winkel, um den die Scheibe verdreht wird, linear von der Komponente der Mausbewegung in Richtung der jeweiligen Koordinatenachse ab. Bewegt man die Maus zum Beispiel diagonal, so drehen sich beide Sensorscheiben mit. Schiebt man sie dagegen in Richtung der Achsen – also vor und zurück beziehungsweise nach rechts oder links – wird nur eine der Laufrollen angetrieben, da die Drehachse der Kugel senkrecht auf der Drehachse der anderen Laufrolle steht. Somit überträgt die Kugel keine Bewegung auf diese Rolle.
Die Drehbewegung der Kugel ist nun in zwei Komponenten aufgespalten, die in elektrische Signale umzusetzen sind. Optomechanische Mäuse erledigen dies mit Hilfe von Gabellichtschranken, die die Bewegung durch die Öffnungen an den Sensorscheiben registrieren. Dabei erzeugt jede Unterbrechung der Lichtschranke einen Impuls.
Aus dem Durchmesser der Laufrollen und der Anzahl der Schlitze in der Scheibe läßt sich die maximale „echte" Auflösung der Maus bestimmen. Sie gibt an, wieviele Impulse erzeugt werden, wenn man die Maus auf der Tischplatte um ein Inch verschiebt. Bezeichnenderweise nennt man sie auch „Mickeys". Typische Werte liegen bei etwa 200 bis 300 Mickeys pro Inch. Durch eine geschickte Programmierung des Maustreibers lassen sich jedoch wesentlich höhere effektive Auflösungen bis zu 10000 Mickeys pro Inch erzeugen.

Bei der Rollkugelmaus sind die beiden Schlitzscheiben mit jeweils zwei Gabellichtschranken zu erkennen.

Richtung erkannt

Die Impulszahl reicht jedoch nicht aus, um die Bewegung der Maus vollständig zu beschreiben. Um die Information über die Richtung ebenfalls aus der Bewegung der Schlitzscheibe zu gewinnen, bedient man sich der sogenannten Quadratursignale. Dazu ist eine zweite Gabellichtschranke erforderlich, die so angeordnet ist, daß die erzeugte Impulsfolge jener der ersten Lichtschranke entspricht, aber zusätzlich um 90 Grad phasenverschoben ist. Dadurch entsteht eine Unsymmetrie bezüglich der Zeitumkehrung. Dreht sich die Schlitzscheibe nun in eine andere Richtung, entsteht aus einer Phasenverschiebung von +90 Grad eine von −90 Grad und umgekehrt.
Zur Drehrichtungsbestimmung könnte man eine Auswerteelektronik verwenden, die zum Beispiel auf die ansteigenden Flanken der Impulsfolgen reagiert. Je nachdem, in welche Richtung sich die Schlitzscheibe dreht, kom-



Schematischer Aufbau einer Rollkugelmaus

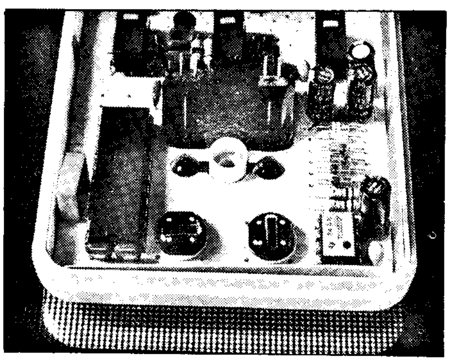
men die Impulsflanken der zweiten Lichtschranke immer ein wenig früher oder später an als die der ersten.
Obwohl das optomechanische Prinzip heute nicht zuletzt wegen der berührungslosen Bewegungsabtastung durch Lichtschranken am weitesten verbreitet ist, arbeiten einige Mäuse wie die Microsoft-Maus (siehe Seite 193) rein mechanisch, oder besser: elektromechanisch.
Hier wird die Impulserzeugung nicht optisch durch Gabellichtschranken mit optischen Unterbrecherscheiben vorgenommen, sondern mit Unterbrecher- und Schleifkontakten. Dadurch stehen die Signale direkt zur Verfügung, die Kontakte sind jedoch einem mechanischen Verschleiß unterworfen, was die Lebensdauer verkürzt. Ansonsten arbeiten mechanische Mäuse nach demselben Prinzip wie die optomechanischen Gegenstücke.

Tor zur Welt

Die Auswertung der Impulsfolgen der insgesamt vier Impulsgeber (Gabellichtschranken oder elektrische Unterbrecher mit Schleifkontakten) sowie der meist zwei bis drei Maustasten überläßt man heutzutage entweder einem Drehgeber mit integrierter Elektronik, die dann über einen Zählimpuls und einen Drehrichtungsausgang verfügt, oder man benutzt gleich einen maskenprogrammierten Einchip-Mikrocontroller, der sich in das Mausgehäuse einbauen läßt. Bei dieser Gelegenheit kümmert er sich auch gleich um den Datenaustausch mit dem angeschlossenen Rechner, da die mühsam gewonnenen Daten über die Bewegung der Maus ja noch mehr oder weniger sinnvoll verwertet werden sollen.
Für die Kommunikation mit dem Rechner bietet sich die serielle Schnittstelle an, da sie ohnehin meist zur Verfügung steht und mit einem Minimum an Verbindungsleitungen auskommt. Früher hat man die Mäuse öfter so konstruiert, daß die aufbereiteten Impulsfolgen zum Rechner übertragen und erst dort weiterverarbeitet wurden. Bei der Microsoft-Bus-Maus ist das auch heute noch der Fall. In der Regel findet die Datenverarbeitung aber auch bei den sogenannten Bus-Mäusen schon in der Maus statt. Oft handelt es sich dabei um eine normale serielle Maus. Die Bus-Karte stellt dann eigentlich nur eine zusätzliche serielle Schnittstelle zur Verfügung. Daher ist ihr Einsatz vor allem dann sinnvoll, wenn die seriellen Schnittstellen bereits von anderen Geräten wie Modem, Plotter oder Drucker belegt sind.
IBMs PC/2-Modelle sind neben dem Tastaturanschluß mit einer zweiten Buchse für ein zusätzliches Eingabegerät ausgerüstet. Dabei handelt es sich ebenfalls um eine serielle Schnittstelle, die jedoch im Gegensatz zu COM1 und COM2 nicht asynchron, sondern synchron arbeitet. Schließt man eine PS/2-Maus dort an, wird der Takt nicht in der Maus, sondern vom Keyboard-Controller erzeugt. An dieser Buchse läßt sich per Adapter auch eine übliche serielle Maus anschließen. Ein derartiger Adapter ist zum Beispiel im Lieferumfang der Microsoft-Maus enthalten.
Einige wenige Hersteller bieten außerdem Versionen für die parallele Schnittstelle an. Die Signale lassen sich hier einzeln zum Rechner übertragen, so daß neben den zwei Pins für +5 Volt und Masse jeweils zwei Kanäle für jede Bewegungsrichtung der Maus belegt sind. Je nach Anzahl der Maustasten werden diese Informationen über ein bis drei weitere Pins herangeführt.

Ohne Mechanik

Gegenüber optomechanischen Mäusen arbeiten optische Mäuse ohne Rollkugel und kraftschlüssige Bewegungsübertragung auf die Sensoren. Dafür benötigen sie aber ein spezielles Maus-Pad, das mit einem einoder mehrfarbigen Muster gedruckt ist. Diese strukturierte Oberfläche übernimmt hier die Funktion der Schlitzscheibe bei optomechanischen Mäusen.
Obwohl sich die Funktionsweisen verschiedener optischer Mäuse im Detail unterscheiden, weisen sie genug prinzipielle Gemeinsamkeiten auf, um das Prinzip sowie die Ähnlichkeit zu optomechanischen Mäusen zu veranschaulichen. Insbesondere arbeiten optische Mäuse nur selten auf den Pads von Fremdfabrikaten.
Das Maus-Pad ist in diesem Beispiel mit einem zweifarbigen Linienraster auf einem silbern reflektierenden Hintergrund bedruckt.

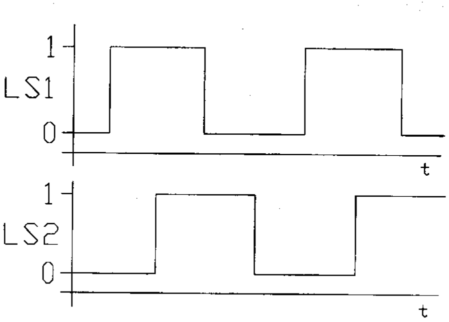


Bei der optischen Maus sind im Vordergrund die Fotodetektoren zu erkennen, in den Öffnungen der Platine zwei kugelförmige Linsen. Direkt dahinter liegen die beiden Leuchtdioden.

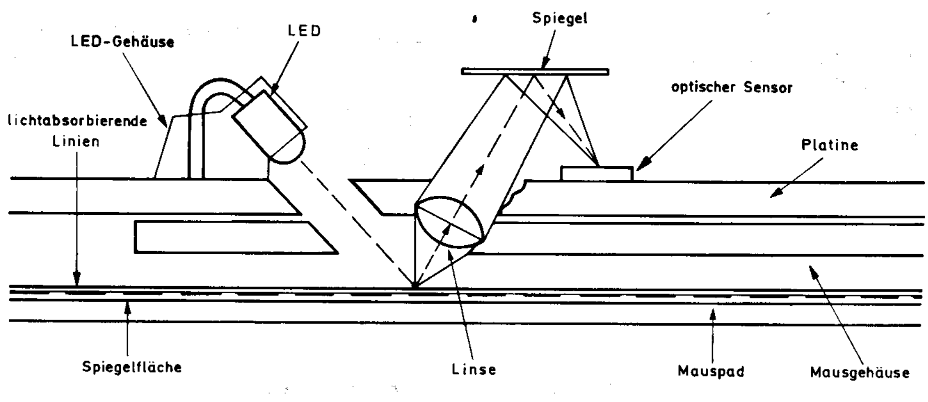
Das Linienraster besteht aus parallelen blauen Linien, auf denen schwarze beziehungsweise grau erscheinende Linien senkrecht stehen. Beide Liniensorten sind mit transparenter Farbe gedruckt. Dabei entspricht der Abstand der Linien voneinander in etwa ihrer Breite, so daß ein Raster aus Quadraten entsteht, die entweder farblos reflektieren oder in blau, schwarz oder in blau und schwarz übereinander gedruckt sind.
Dieses Muster wird von einer Sensoranordnung in der Maus abgetastet. Durch zwei Öffnungen an der Unterseite der Maus beleuchten zwei Leuchtdioden die Oberfläche des Pads mit rotem beziehungsweise infrarotem Licht. Die blau und schwarz erscheinenden Druckfarben sind so beschaffen, daß sie das Licht jeweils einer LED-Farbe gut absorbieren, das Licht der anderen LED aber transparent erscheinen lassen. Auf diese Weise sind im Licht jeder Leuchtdiode jeweils nur die Linien einer Farbe dunkel, während die anderen ähnlich hell reflektieren wie der Hintergrund.
Zwei kugelförmige Linsen, die direkt an der Öffnung auf der Unterseite der Maus angeordnet sind, projizieren das vom Pad reflektierte Licht (mithin das Linienmuster des Pads) über Umlenkspiegel im Innern der Maus auf jeweils einen optischen Sensor. Dieser besteht aus einem linearen Array mit vier Fotodioden. Ihre Abmessungen und Abstände sind so bemessen, daß die Gesamtlänge der vier Sensorzellen dem Linenabstand in dem projizierten Bild entspricht. Es wird also gerade eine Gitterperiode auf die vier Zellen des Arrays abgebildet, wobei jede Zelle zu ihrer Nachbarzelle eine Viertelperiode (oder 90 Grad Abstand) aufweist. Darüber hinaus stehen die Längsachsen der beiden Arrays senkrecht aufeinander.
Da die beiden verschiedenfarbigen Liniensorten ebenso wie die Achsen der Detektor-Arrays in der Bildebene senkrecht aufeinander stehen, nimmt der eine Sensor nur die waagerechten, der andere nur die senkrechten Linien wahr. Auf diese Weise wird die Bewegung der Maus so ähnlich wie bei einer Rollkugelmaus in Komponenten zerlegt.

Richtig gesehen

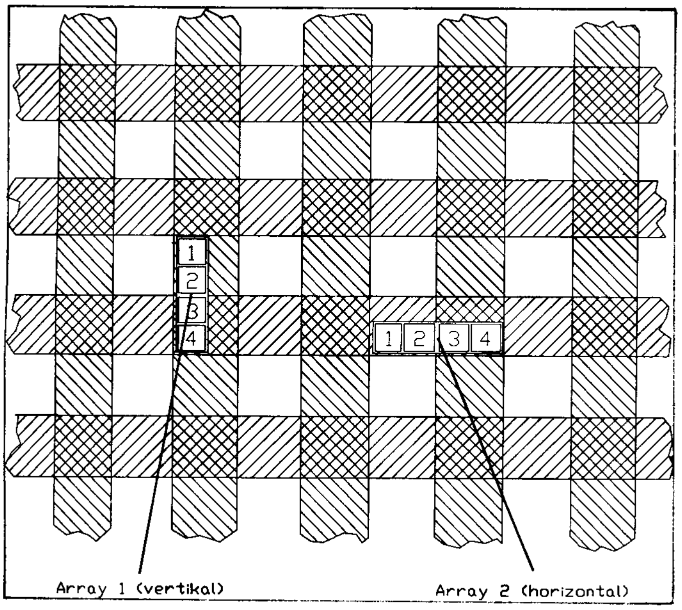
Bei der Mausbewegung wandern die projizierten Linien über das Array. Entsprechend liefert jede Zelle ein Signal, das wegen der Geometrie des Detektors jeweils um 90 Grad gegenüber dem Signal der Nachbarzelle verschoben ist. Diese Signalfolgen entsprechen aber genau denen von optomechanischen Mäusen.
Nun scheinen aber doppelt so viele Sensoren vorhanden zu sein, wie vom Prinzip her eigentlich notwendig wären. Die Projektion von



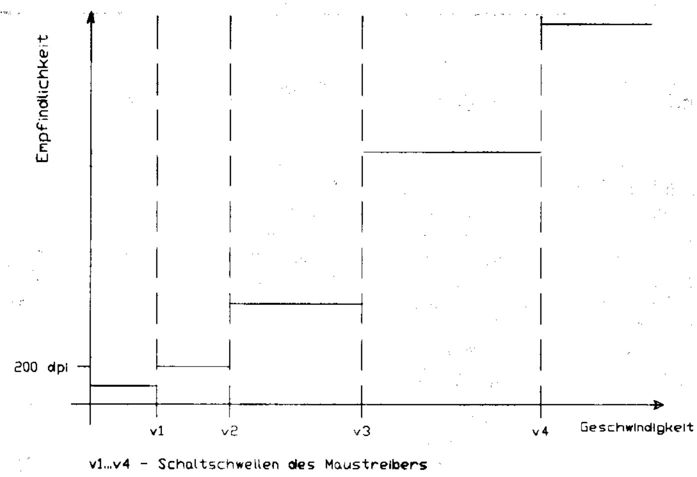
Durch eine Phasenverschiebung der beiden Impulsfolgen um 90 Grad läßt sich die Drehrichtung der Kugel bestimmen.



Schnittzeichnung einer optischen Maus



Die Maussensoren sind so angeordnet, daß auf die vier Zellen eine Gitterperiode abgebildet wird.



Je nach Geschwindigkeit der Bewegung erhöhen nichtlineare dynamische Mäuse die Empfindlichkeit an definierten Schaltschwellen.

Linien verschiedener Transparenz über eine einfache Optik und einen Umlenkspiegel ist jedoch nicht so kontrastreich und störungsarm wie zum Beispiel das Signal, das die Gabellichtschranken einer optomechanischen Maus liefern. Das liegt unter anderem an nicht idealen Eigenschaften der Druckfarben oder einem schlechteren Reflexionsvermögen des Pads, wenn es zum Beispiel verschmutzt ist.
Deshalb wendet man einen Kunstgriff an und wertet nicht das Signal einer einzelnen Sensorzelle aus, sondern die Differenz der Signale zweier Zellen, die untereinander eine Phasenverschiebung von 180 Grad (also eine halbe Periode) haben. Dadurch ist ein maximaler Kontrast des projizierten Musters gewährleistet, denn eine der beiden Zellen tastet immer eine Linie und die andere einen Zwischenraum ab. Da nur die Differenz der Signale verarbeitet wird, spielt die absolute Helligkeit keine Rolle, solange nur genügend Kontrast vorhanden ist.

Großer Aufwand

Warum machen sich die Hersteller aber die Mühe, ein so aufwendiges optisches System zu konstruieren und Störeinflüsse fernzuhalten, wo doch optomechanische Mäuse robuster und zudem billiger zu sein scheinen? Überdies ist ein spezielles Pad notwendig, ohne das man die Maus außer Funktion setzt. Eine optomechanische Maus kann man dagegen auf jeder Oberfläche benutzen, auch auf einem verstaubten Aktenstapel. Allerdings verhindert eine gut haftende Kugeloberfläche nicht nur Schlupf, sonder fördert auch Schmutz ins Mausinnere. Dies ließe sich zwar durch konstruktive Maßnahmen vermindern, doch je nach Belastung der Umgebung ist von Zeit zu Zeit eine Reinigung der Maus erforderlich.
Das Säubern der Kugel ist noch ein recht leichtes Unterfangen, da sie sich leicht herausnehmen läßt. Schwieriger wird es schon bei den Laufrollen, auf denen sich der Schmutz der Kugel mit der Zeit ablagert. Wenn sich aber Fasern oder Haare mit klebrigen Bestandteilen wie Cola um die Wellen wickeln, verlangt dies besonderes Geschick.
Bei einer optischen Maus sammelt sich der Schmutz fast nur an den Filzstreifen an, auf denen sie über das Pad gleitet. Dies hat jedoch keinen Einfluß auf die Bewegungssensoren, so daß sie nur etwas schwerer gleitet.

Schaltgetriebe ...

Bei den meisten Mäusen ist die Bewegung der Maus und des Zeigers auf dem Bildschirm fest gekoppelt. Manchmal wäre es jedoch komfortabler, wenn sich die Kopplung aufheben ließe. Zum Beispiel benötigt man in einer CAD-Anwendung beim Zeichnen von Details eine höhere Präzision als beim schnellen Positionieren des Zeigers zu einem Menü. Bei einer Auflösung von 800 × 600 Pixeln muß man im Extremfall eines von 480 000 Pixeln treffen. In solchen Fällen ist es komfortabler, wenn sich die Empfindlichkeit der Maus umschalten läßt und sich dadurch die Anzahl der gesendeten Mickeys pro Inch ändert. Einen Schalter zum hardwaremäßigen Umschalten findet man jedoch nur selten bei Mäusen. Häufiger trifft man dagegen Maustreiber an, die über dieses Feature verfügen.
Eine noch komfortablere Lösung beruht auf der Tatsache, daß die augenblicklich gewünschte Empfindlichkeit und die momentane Geschwindigkeit der Maus stark korrelieren. Das bedeutet, daß der Benutzer in der Regel eine relativ schnelle Handbewegung ausführt, um den Zeiger quer über den Bildschirm zu bewegen. Beschäftigt er sich dagegen mit Details einer Grafik, so verschiebt er die Maus eher langsam, um ein Pixel möglichst präzise ansteuern zu können. Eine derartige Maus würde ihre Empfindlichkeit automatisch je nach der Bewegungsgeschwindigkeit anpassen.

... oder Automatik

Nur wenige Hersteller haben dies jedoch hardwaremäßig gelöst. Sie liefern aber in letzter Zeit zunehmend Maustreiber mit aus, die diese automatische (beziehungsweise dynamische) Anpassung erlauben, zumal sie sich leicht realisieren läßt. Sendet die Maus zum Beispiel die Daten über die Positionsänderung seit der letzten Übertragung in konstanten Zeitabständen, so hängt der Wert der relativen Ortsänderung linear von der Geschwindigkeit der Maus ab. Ausgehend von dieser Information sind Schwellenwerte für die Geschwindigkeit definiert, oberhalb derer die Anzahl Mickeys vor der Weiterleitung an das Anwenderprogramm mit einem Faktor multipliziert wird. Dadurch lassen sich sehr hohe effektive Empfindlichkeiten von bis zu 10 000 Mickeys pro Inch realisieren.
Die Mausbewegung erfolgt dann zwar ruckhaft, der Benutzer merkt dies jedoch kaum, da die Maus bei Verlangsamung der Bewegung automatisch „herunterschaltet“ und dadurch feinfühlig wird. Es bleibt allerdings kritisch anzumerken, daß die Arbeit mit dynamischen Mäusen sehr nervtötend sein kann, wenn die nichtlineare Kennlinie falsch gewählt ist. Der Grad der Dynamik sollte sich daher wählen oder notfalls auch ganz abschalten lassen.

Auswahl leichtgemacht

Ob man sich beim Kauf nun für eine mechanische, eine optomechanische oder eine optische Maus entscheiden sollte, hängt unter anderem vom Geldbeutel und dem geplanten Einsatzgebiet ab. Der Hobby-Anwender, der sie vor allem zur komfortableren Bedienung der Textverarbeitung braucht und gelegentlich eine Skizze mit einem Zeichenprogramm erstellt, wird sich wohl im riesigen Angebot an Rollkugelmäusen umsehen.
Wie der Vergleichstest in c't 12/89 zeigt, erhält man bereits für unter hundert D-Mark Exemplare, die gut in der Hand liegen und mit denen sich gut arbeiten läßt. Auf glatten Tischoberflächen muß man zwar öfters ansetzen, bis sich die Kugel dreht und den Zeiger auf dem Bildschirm bewegt, ein Maus-Pad aus Filz oder ähnlichem schafft hier jedoch Abhilfe. Nicht selten ist es bereits im Lieferumfang enthalten.
Optische Mäuse findet man vor allem im professionellen Bereich, beispielsweise in Konstruktionsbüros. Da sie die Bewegung anhand des Linienrasters auf dem Pad registrieren, erlauben sie eine sehr hohe Präzision. Wer häufig mit CAD-Software Zeichnungen oder Pläne erstellt, sollte eine optische Maus trotz des hohen Preises (über 400 D-Mark) zumindest ins Auge fassen.

Literatur

/1/ Sol Sherr, Input Devices, Academic Press 1988
/2/ S. T. Kirsch, Detector for Electro-Optical Mouse, US-Patent Nr. 4 546 347, 1985
/3/ S. T. Kirsch, Electro-Optical Mouse, US-Patent Nr. 4 546 347, 1982

Nachdruck aus c't – Magazin für Computertechnik, Heft 12/1989

JPI TopSpeed Modula-2

Nach seinem guten Abschneiden beim Compiler-Wettlauf in MP 4/90 verdient es der TopSpeed Modula Compiler, einmal genauer vorgestellt zu werden. Dabei geht es weniger um die Sprache Modula, als vielmehr um die Qualitäten des Compilers und der Entwicklungsumgebung.

Die Entwicklungsumgebung

In vielem knüpft JPI Modula an die guten Eigenschaften von Turbo-Pascal an. Kein Wunder: Die Firma Jensen & Partners International (nichts anderes heißt JPI) wurde von ehemaligen Borland-Programmierern in England gegründet. Und in der Geschwindigkeit hat man das Vorbildprodukt bereits übertroffen. Die Menütechnik erinnert durchaus an Borland-Produkte (Menübäume statt Dialogboxen wie bei Microsoft), darüber hinaus können die Menüs aber völlig frei konfiguriert werden. Das betrifft natürlich die Farben, die ebenso interaktiv zu jedem beliebigen Zeitpunkt (bei Turbo nur mit TINST) geändert werden können wie die Anordnung auf dem Bildschirm. Das Hauptmenü im Bild oben habe ich in die rechte obere Ecke gerückt, damit es den Quelltext nicht verdeckt – ursprünglich stand es links. Darüber hinaus lassen sich aber auch die einzelnen Menüpunkte und die Hotkeys festlegen. Verantwortlich dafür ist das Menüfile M2.MNU, von dem ein Ausschnitt im Bild darunter dargestellt ist. Durch Einrücken erfolgt automatisch die Zuordnung zu Untermenüs, die Optionen *Load Files, Pick Files* usw. bilden also ein Untermenü von *Files*. Dieser Menüaufwand wird dann (im oberen Bild) rechts zu erkennen sein. Der Buchstabe in geschweiften Klammern wird später hervorgehoben dargestellt (heller oder andersfarbig) und kann im entsprechenden Menü zur direkten Aktivierung dieses Punktes verwendet werden (als Alternative zur Bewegung des Farbbalkens mit den Cursortasten). In spitzen Klammern sind die Shortcuts festgelegt. Mit einem einfachen Editor kann so die gesamte Nutzeroberfläche geändert werden. Die Fehlermeldungen in einem extra ASCII-File zu halten gehörte schon bei Turbo 3.0 zum guten Ton, auch die Online-Hilfe ist inzwischen selbstverständlich.

Dem Programmierer steht ein Multi-Window-Editor (Wordstar-kompatibel, aber auch beliebig anders einzurichten) zur Verfügung, mit dem bis zu vier Dateien gleichzeitig bearbeitet werden können (siehe Bild 1). Beim Verlassen der Entwicklungsumgebung werden nicht nur die gerade geladenen Quelltexte, sondern auch die aktuelle Cursorposition, geöffnete Fenster etc. in ein sogenanntes Session-File gespeichert, das es ermöglicht, genau an der Stelle weiterzumachen, wo man beim letzten Mal aufgehört hat. Die Auto-Save-Option garantiert, daß vor dem Lauf eines Programms (und dem damit möglichen Absturz) sowie beim Verlassen stets alle geänderten Quelltexte gesichert werden, was insbesondere bei mehreren gleichzeitig in Bearbeitung befindlichen Texten von Bedeutung ist.

Eine weitere nützliche Einrichtung ist das Redirection File, mit dem sich der Compiler sein eigenes Environment anlegt. Damit werden nicht nur getrennte Directories für Quelltexte, Bibliotheken, Objete etc. festgelegt; es kann auch genutzt werden, um lokale Festlegungen im Netzbetrieb zu treffen.

Der Compiler

Compiler und Linker können wahlweise aus der integrierten Entwicklungsumgebung oder als Kommandozeile aufgerufen werden, ohne daß dafür ein extra Programm nötig wäre (wie TPC bei Turbo-Pascal); es genügt die Angabe eines Schalters. Die Übersetzung der Implementationsteile kann in beliebiger Reihenfolge vorgenommen werden, unaktuelle Definitionsmodule gibt es nicht, da sie stets neu compiliert werden – die Geschwindigkeit des Compilers (der Hersteller gibt 20000 Zeilen pro Minute an) läßt das zu. Gleichzeitig entfallen die Symbol-Files, weil die Definitionsmodule direkt ins Programm übersetzt werden. Große Projekte werden durch die Make-Funktion unterstützt. Der Compiler erzeugt in einem Paß Microsoft-kompatiblen Objektcode – angesichts der großen Geschwindigkeit eine erstaunliche Leistung. Der Programmierer besitzt damit eine Schnittstelle zu anderen Sprachen, auch zum Assembler. Voraussetzung ist aber, die Definition der anderssprachigen Module in einem Modula-DEF-File aufzuführen und beim Linken die richtige Reihenfolge der Objektmodule einzuhalten.

Der Linker arbeitet – wie an der Codegröße erkennbar und von Turbo-Pascal gewöhnt – intelligent, das heißt, er entfernt automatisch nicht aufgerufene Funktionen. Außerdem genügt beim Linken die Angabe des Hauptmoduls, der Informationen über alle weiteren benötigten Objekte (in einem Include-Record) und die erforderliche Aktualität (in einem Version-Record) enthält.

Bei Nutzung des Microsoft-Linkers zur Einbindung fremder Objekte werden wesentlich größere EXE-Files erzeugt, und alle Files müssen explizit und in der richtigen Reihenfolge angegeben werden.

In zwei getrennten Bibliotheksmodulen sind zum einen der Sprachstandard nach Wirth und zum anderen die pragmatischen Erweiterungen von JPI implementiert. Obwohl zur Sprache Modula gehörend, wird Coprocessing nicht von allen Modula Compilern auch wirklich angeboten. JPI offeriert hier einen Time-Slice-Scheduler für echtes Multitasking. Die OS/2-Version setzt im Protected Mode Koprozesse in separate OS/2-Threads um. Leider gibt es für MS-DOS keinen Schalter zur Nutzung der Vorzüge des 80286. Der Arithmetik-Koprozessor i8087 wird unterstützt und – wenn nicht vorhanden – stets emuliert.

Die Geschwindigkeit der Programme ist beeindruckend und liegt in Bereichen, die man sonst nur C-Compilern zugetraut hätte (siehe /1/). Sie wird durch eine ausgefeilte Optimierung (z. B. Registervergabe) erreicht.

```
#1 C:\PIT\JPI\SIEVE.MOD                              Insert   Line 15    Col 1
MODULE Sieve;                                                   JPI Modula-2
FROM IO IMPORT WrLn, WrInt, WrStr;                              Files
CONST Size=8190;
VAR Flags: ARRAY[0..Size] OF BOOLEAN;                            Load file
    i, prime, k ,count, iter: CARDIN ============ Load file ==============
                                   Window 1: sieve.mod
BEGIN                              Window 2: *.MOD
  WrLn; WrStr("100 iterations");   Window 3: *.MOD
  FOR iter:= 1 TO 100 DO           Window 4: fibo.mod
      count:=0;
      FOR i:=0 TO Size DO Flags[i]:= TRUE                        Dos shell
      END;                                                       Execute
      FOR i:=0 TO Size DO                                        Quit
#4 C:\PIT\JPI\FIBO.MOD                               Insert   Line 1     Col 1
MODULE fibo;
FROM IO IMPORT WrLn, WrStr, WrInt;
VAR
i,k:INTEGER; c: CHAR;

PROCEDURE fib(n : INTEGER): INTEGER;
BEGIN

↵-select  Esc-cancel
```

Bild 1 Die integrierte Entwicklungsumgebung mit Multiwindow-Editor

```
! Pop Up [ JPI Modula-2 ]           |                    <F10>
  {F}iles                           |                    <AltF>
    {L}oad file                     | Load file          <F3>
    {P}ick file                     | Pick file          <AltF3>
    {S}ave file                     | Save file          <F2>
    {A}ll save                      | Save all files
    {M}ain module                   | Main module
    {C}hange dir                    | Change dir
    {F}iles dir                     | Directory
    {D}os shell                     | DOS shell          <AltD>
    {E}xecute                       | Execute
    {Q}uit                          | Quit               <AltX>
  {E}dit                            | Editor             <AltE>
  {C}ompile                         | Compiler           <AltC>
  {M}ake                            | Make               <AltM>
  {L}ink                            | Linker             <AltL>
  {R}un                             | Run program        <AltR>
  {O}ptions                         |                    <AltO>
```

Bild 2 Ausschnitt aus der Menüdatei

Gut unterstützt JPI – und das sei extra erwähnt, weil mit diesem Argument häufig eine Lanze für C gebrochen wird – die hardwarenahe Programmierung. Vieles ist von Turbo bekannt: Der Aufruf von DOS- und BIOS-Funktionen, der direkte Zugriff auf Speicherzellen und Ports sowie die Erzwingung von FAR-Sprüngen zur Einbindung externer Programmteile. Darüber hinaus existieren beim TopSpeed-Compiler Schalter, mit denen festgelegt werden kann, welche Register bei Eintritt in eine Prozedur gerettet werden sollen, ob mit RET oder IRET beendet wird, welcher Programmcode stets zu Beginn und am Ende einer Prozedur eingefügt werden soll (wie für MS-Windows-Applikationen gefordert) bzw. ob statt dessen nutzereigene Interrupts angesprungen werden sollen (zum Tracen nutzbar).

Für konkurrierende Prozesse kann es nötig sein, die Optimierung des Gebrauchs von Variablen abzuschalten, die automatisch so lange wie möglich in Registern gehalten werden (keine Speicherklasse REGISTER wie in C nötig), was unter Umständen bedeutet, daß eine Speicherzelle im gemeinsamen Zugriff nicht aktuell ist. Per Schalter kann das Auslagern in den Speicher nach jeder Operation erzwungen werden.

Über die Annehmlichkeiten von Turbo-Pascal hinaus – die man fast sämtlich findet – sind mir noch Funktionen zur Adreßrechnung (AddAddr, IncAddr), zum Sperren der Hardwareinterrupts (DI, EI) – wofür man in Turbo Inlines bemühen muß – und eine gute Mathematik-Bibliothek aufgefallen. Den professionellen Window-Modul findet man sonst nur in Toolboxen.

Die mitgelieferten Bibliotheken und Tools sind nicht so umfangreich wie beim Logitech-Compiler, aber eigentlich bleiben keine Wünsche offen. Im einzelnen besteht die Bibliothek aus 12 Modulen (plus SYSTEM), deren Namen aber – ebenso wie die vieler Prozeduren – vom Standard abweichen (z. B. IO statt InOut). Im Überblick:

- **SYSTEM** konkurrierende Prozesse, Speicher- und Portzugriff
- **AsmLib** Hilfsmodul, der die in Assemblergeschriebenen Bibliotheksteile zusammenfaßt
- **MATHLIB** trigonometrische, hyperbolische, logarithmische Funktionen, 8087-Support
- **Str** String-Operationen (Verketten, Vergleichen, Umwandeln)
- **Lib** Sammel-Modul: Sortieren, Zufallszahlen, DOS-Zugriff, Adreß-Operationen, Fehler-Handling, Sound
- **Storage** Vergabe/Freigabe von Speicher, Informationen über Verfügbarkeit
- **Prozess** Starten und Synchronisieren von Prozessen
- **Graph** Zeichnen von Pixeln, Linien, Kreisen, Füllen von Polygonen (unterstützt CGA, Hercules, EGA, VGA)
- **FIO** Filearbeit (Öffnen, Schließen, Lesen, Schreiben, Suchen, Löschen, Umbenennen)
- **IO** formatierte Ein- und Ausgabe auf Konsole, direkte Kontrolle der Tastatur, Umleitung von Ein- und Ausgabe
- **Windows** Öffnen, Schließen, Verschieben, Schreiben in Fenster, Setzen von Rahmen, Farbe, Titel, Cursor

Fortsetzung auf Seite 211

Lisp

Teil 1

Volker Bauche, Dr. Horst Friedrich, Berlin

Die Programmiersprache Lisp (Abkürzung für List Processing) wurde bereits Ende der 50er Jahre etwa zeitgleich mit Fortran entwickelt und zählt somit zu den ältesten Programmiersprachen überhaupt. Sie diente ursprünglich hauptsächlich der Symbolmanipulation, überstreicht aber heute bereits alle Anwendungsgebiete. Lisp ist im Gegensatz zu den sequentiellen und imperativen Sprachen wie Fortran, Pascal, Algol und C eine funktionale Sprache, die in ihrer Philosophie stark von anderen Sprachen abweicht und durch ihre Klarheit und ihr einheitliches Konzept besticht. Das soll nicht heißen, daß Lisp all diese Sprachen ersetzen soll, aber man sollte diese hochentwickelte Sprache bei der Auswahl der Programmiersprache besonders für anspruchsvollere Aufgaben mit in Erwägung ziehen. Viele in den anderen genannten Sprachen aufwendig umzusetzende Programmiertechnologien lassen sich in Lisp sehr viel effektiver anwenden. Dazu zählen unter anderem Methoden der Künstlichen Intelligenz, etwa der Wissensrepräsentation oder des Wissenserwerbs. Ein großer Vorteil der Sprache liegt in der später noch ausführlicher zu erläuternden formalen Nicht-Unterscheidbarkeit von Daten und Programmen in Lisp.

Internationale Untersuchungen haben ergeben, daß sich etwa 60 Prozent aller Lisp-Anwender mit Expertensystemen beschäftigen. Typische Einsatzgebiete sind die Planung (26 % der Lisp-Anwender), die Softwareentwicklung (21 %), die Interpretation von Signalen (17 %), die Robotertechnik (17 %), das Gebiet des CAD/CAE (16 %), die Verarbeitung natürlicher Sprache (14 %), die Diagnose (12 %), das Training und die Simulation (12 %) und die industrielle Automatisierung (11 %), wobei sich die Einsatzgebiete überschneiden (z. B. Robotertechnik und Planung).
In diesem Kurs werden Sie Beispiele kennenlernen, in denen besonders die Vorteile der Sprache sichtbar werden. Als Test könnten Sie versuchen, das eine oder andere vorgestellte Beispiel in einer anderen Programmiersprache Ihrer Wahl zu schreiben.

Common Lisp entstand Anfang der 80er Jahre, nachdem sich Lisp in viele Dialekte aufgespalten hatte, und wird heute als Quasi-Industriestandard akzeptiert. Weltweit arbeiten mehrere Gruppen an einer weiterführenden Lisp-Standardisierung.

Als Grundlage für diesen Kurs wurde eine Teilmenge von Common Lisp gewählt. Eine vollständige Implementation von Common Lisp wurde am Zentralinstitut für Kybernetik und Informationsprozesse der AdW vorgenommen und ist unter der Bezeichnung EXPERT Common LISP (XCL) für den K 1840 und für den P 8000 verfügbar. Diese Implementation ist für den K 1840 über Robotron-Projekt Dresden, oder über das ZKI der AdW der DDR zu beziehen. Die Version für den P 8000 ist nur über das ZKI erhältlich. Alle angeführten Beispiele können jedoch auch unter Golden Common LISP von Gold Hill Computers, mit einigen Namensänderungen unter PC-Scheme (sprich: Skiem) von Texas Instruments und ein großer Teil von ihnen unter MuLisp von Microsoft abgearbeitet werden. Diese Systeme laufen auch auf dem EC 1834, XT- und AT-Rechnern. Für diese Rechner existieren außerdem die Systeme TLC-LISP (von The Lisp Company) und IQ-LISP (von Integral Quality). Das System X-LISP ist in C geschrieben und als Public-domain-Software im Quelltext verfügbar. Für den A 7150 ist von den genannten Lisp-Systemen Golden Common Lisp (Gold Hill Computers) verfügbar. Das MuLisp für U 880-Rechner bzw. für den A 7100 unterscheidet sich dagegen sehr stark von dem hier vorgestellten Lisp-Dialekt.

In diesem Lisp-Kurs werden Übungen und Aufgaben angegeben. Zum besseren Verständnis von Lisp sollten Sie die Übungen mit einem Lisp-System am Computer abarbeiten. Die Lösungen der Aufgaben finden Sie am Ende des Teils, in dem sie gestellt wurden.

Symbolische Ausdrücke

Die symbolischen Ausdrücke oder *S-Ausdrücke* sind die grundlegenden Datenstrukturen in Lisp. Sie können einerseits als Daten und andererseits als Prozeduren interpretiert werden. Da beide formal nicht unterscheidbar sind, eröffnet sich dem Lisp-Nutzer die Möglichkeit, scheinbar komplizierte Probleme mit verblüffender Einfachheit zu kodieren. So kann ein Programm andere ohne weiteres als Daten benutzen, also auch erzeugen, verändern und aktivieren.

Lisp stellt eine Vielzahl von Datentypen bereit. Im Gegensatz zu konventionellen Programmiersprachen wird in Lisp nicht den Variablen, sondern den Daten ein Typ zugeordnet, so daß eine Lisp-Variable beliebige Daten aufnehmen kann. Das Typsystem basiert auf einer Untermengenhierarchie, in die sowohl alle vordefinierten als auch die vom Nutzer definierten Typen eingeordnet werden.

Atome

Von den atomaren Datentypen werden hier nur Symbole, ganze Zahlen, Gleitkommazahlen, Zeichenketten und in einem späteren Lehrgang Strukturen vorgestellt. Common Lisp enthält außerdem noch Zeichen, Vektoren, Felder, Hash-Tabellen, Pakete, Ströme, Pfadnamen, Gleitkommazahlen unterschiedlicher Genauigkeit, gebrochene Zahlen, komplexe Zahlen sowie Zufallszahlen.

Ganze Zahlen (Integer) sind Aneinanderreihungen von Ziffern eventuell auch mit einem führenden Vorzeichen (+ oder −), so beispielsweise
−0, 0, +6, 1024 *und*
15511210043330985984000000
(die Fakultät von 25).

Gleitkommazahlen (Float) bestehen aus einer Folge von Ziffern, die einen Dezimalpunkt enthält. Sie können mit einem Vorzeichen und mit einem Exponenten notiert werden. Beispiele für Gleitkommazahlen sind:
2.0, +6.02e23,
3.1415926535897932384 (pi).

Zeichenketten sind beliebige Zeichenfolgen, die in zwei Anführungszeichen (") eingeschlossen werden, beispielsweise "z", "Zeichenkette" und "Common LISP".

Symbole sind Aneinanderreihungen von Buchstaben, Zahlen und Sonderzeichen (mit Ausnahme der Zeichen **(**, **)**, **'**, **'**, **"**, **;**, *Komma* und *Leerzeichen*). Die Symbole sind mit den Bezeichnern anderer Programmiersprachen vergleichbar. Lisp-Symbole sind zum Beispiel: X, atom, +, −, NIL, K17, 10-B (Man beachte: −1 ist eine Zahl und 1− ein Symbol). Das Lisp-Symbol wandelt alle Kleinbuchstaben aus Symbolnamen in Großbuchstaben um, so daß x und X dasselbe Symbol bezeichnen.

Listen

Eine Liste wird in Lisp durch das Einschließen von Objekten in eine öffnende und eine schließende Klammer notiert. Da Listen selbst wieder Objekte von Lisp sind, gibt es somit Listen, die Listen enthalten.

(1 2 3) eine Liste mit den Elementen 1, 2 und 3
(a (1 3) b) eine Liste mit den Elementen a (ein Symbol), (1 3) (eine Liste) und b (ein Symbol)

Eine spezielle Liste ist die leere Liste. Sie kann durch eine öffnende und eine schließende Klammer, also **()**, oder durch das Symbol NIL notiert werden. Das Semikolon kennzeichnet den Beginn eines Kommentars, der durch das Zeilenende begrenzt ist.

Die erste Lisp-Sitzung

Ehe man beginnt, in einem Lisp-System zu arbeiten, sollte man sich dessen grundsätzliche Arbeitsweise klarmachen. Ein Lisp-System arbeitet interpretativ: Ein Lisp-Ausdruck (S-Ausdruck) wird eingelesen, ausgewertet (der Wert des Lisp-Ausdrucks berechnet) und anschließend das Ergebnis ausgegeben. Wird durch den Lisp-Interpreter ein Fehler entdeckt, gerät man in einen sogenannten Break-Status. Man sollte vor dem Arbeiten in der Dokumentation nachlesen, wie man in diesem Falle im jeweiligen System weiterarbeiten kann (z. B. durch Eingabe von :r in XCL, t in MuLisp, <CTRL>C in GC-LISP, <CTRL>Q in PC-Scheme).

Nachdem das Lisp-System gestartet wurde, zeigt es mit einem Promptzeichen (hier das Zeichen >) seine Eingabebereitschaft an:

Starten des Lisp-Systems

>	Ausgabe des Promptzeichens
>**3**	Eingabe des Objektes 3, dieses wird ausgewertet
3	Ausgabe des Ergebnisses der Auswertung

>() nun Eingabe einer leeren Liste
NIL Ergebnis der Auswertung
Beenden der ersten Lisp-Sitzung

Zahlen werden als Konstanten betrachtet und liefern bei Auswertung sich selbst als Wert. Konstanten sind auch die Symbole T und NIL (das auch als Zeichenfolge **()** eingebenen werden kann). NIL wird immer zu NIL und T immer zu T ausgewertet.

Übung 1
In der ersten Lisp-Sitzung sollten Sie sich durch die Eingabe von Zahlen, Zeichenketten und der Symbole NIL und T mit der Arbeitsweise des Interpreters vertraut machen. Provozieren Sie durch Eingabe der Liste (1 22 33) einen Fehler, damit Sie in den Break-Zustand des Lisp-Systems gelangen, und verlassen Sie diesen anschließend. Mit (exit) oder (system) können Sie die Lisp-Sitzung beendet werden.

Formen und Funktionen

Um richtige Berechnungen (Auswertungen) in Lisp durchführen zu können, muß man Funktionsaufrufe eingeben. Die Syntax eines Funktionsaufrufes ist sehr einfach: Ein Funktionsaufruf ist eine Liste, deren erstes Element das Symbol einer Funktion sein muß (sonst kommt es zu einem Fehler) und deren restliche Elemente passende Argumente für diese Funktion sein müssen. In Common Lisp ist + das Symbol für die Additionsfunktion:

(+ 3 6) ⇒ 9

In allen Beispielen steht die Zeichenfolge ⇒ für „wird ausgewertet zu". Der Aufruf der Funktion + mit den Argumenten 3 und 6 wird also zu 9 ausgewertet. Das Symbol + besitzt eine besondere Eigenschaft: Es ist auch als Funktion verwendbar. Die Argumente eines Funktionsaufrufes werden vor der Aktivierung der Funktion ausgewertet. In der Form (+ (* 2 8) (− 12 8)) wird zuerst (* 2 8) und dann (− 12 8) ausgewertet; die so erhaltenen Ergebnisse (16 und 4) werden der Additionsfunktion übergeben. Anschließend wird das Gesamtergebnis 20 ausgegeben.
Ein auswertbarer S-Ausdruck wird auch als Form bezeichnet. Damit man unterscheiden kann, ob eine Liste eine Form oder eine Datenliste ist, wird die Spezialform Quote verwendet. Im Gegensatz zu den Funktionen bleiben bei den Spezialformen die Argumente unausgewertet. Quote erhält ein Argument und liefert als Ergebnis das Argument ohne Auswertung zurück. Somit kann Quote verwendet werden, um die Auswertung zu verhindern.

(quote (+ 3 6)) ⇒ (+ 3 6)
(quote objekt) ⇒ OBJEKT da Lisp alle Buchstaben in Großbuchstaben umwandelt:
Da Konstanten sehr häufig verwendet werden, gibt es für (quote objekt) die abkürzende Schreibweise 'objekt.
'(+ 3 6) == (quote (+ 3 6))
⇒ (+ 3 6)
'(das ist eine liste)
== (quote (das ist eine liste))
⇒ (DAS IST EINE LISTE)
Das doppelte Gleichheitszeichen == ist dabei zu lesen als *ist identisch mit*.

Funktionen zur Listenverarbeitung

Die Grundfunktionen für die Listenverarbeitung sind Car Cdr und Cons. Car liefert das erste Element einer Liste und Cdr die Restliste ohne das erste Element. Die Funktion Cons erhält als Argument ein Objekt und eine Liste und erzeugt eine neue Liste, die aus dem Objekt als erstes Element und aus den Elementen der Liste des zweiten Argumentes besteht.

(car '(eine liste)) ⇒ EINE
(cdr '(eine liste)) ⇒ (LISTE)
(cons 'eine '(liste)) ⇒ (EINE LISTE)
(cdr '(ein-element)) ⇒ NIL

Car und Cdr liefern für die leere Liste, also für NIL, als Ergebnis wiederum NIL.

(cons 'peter nil))
== (cons 'peter ())
⇒ (PETER)
(cons nil nil) == (cons () ())
⇒ (NIL)

Da Funktionen ihre Argumente auswerten, kann man Car Cdr und Cons auch ineinandergeschachtelt verwenden:

(car (cons 'eine '(liste)))
⇒ EINE
(cdr (cons 'eine '(liste)))
⇒ (LISTE)
(car (cdr '(eine liste)))
⇒ LISTE

Man beachte, daß (car '(cdr (eine liste))) das Symbol Cdr als Ergebnis liefert, da es das erste Element der „gequoteten" Liste ist.
Sehr häufig müssen mehrere Car- und Cdr-Funktionen nacheinander ausgeführt werden, um auf ein bestimmtes Objekt einer Liste zugreifen zu können. Dazu können Funktionen der *Car-Cdr-Familie* verwendet werden. Die Namen der Funktionen dieser Familie sind Cxxr, Cxxxr und Cxxxxr, wobei jedes x entweder durch ein a (für car) oder durch ein d (für Cdr) zu ersetzen ist:

(cadr '(a b c)
== (car (cdr '(a b c)))
⇒ B
(cadadr '((a b) (c d) (e f))) ==
(car (cdr (car (cdr '((a b) (c d) (e f))))))
⇒ D

Die Reihenfolge der Anwendung von Car und Cdr ist genau umgekehrt zu der Reihenfolge der Notation von d und a in den Namen der Funktionen.

Übung 2
Durch Anwendungen von Funktionen der CAR-CDR-Familie soll auf das Symbol S in den folgenden Listen zugegriffen werden:
2a) (L I S P)
2b) ((L I) (S P))
2c) (((L) (I) (S) (P)))
2d) (L (I) ((S)) (((P))))
2e) ((((L))) ((I)) (S) P)
2f) ((((L) I) S) P)
Zusätzlich zu den Funktionen der Car-Cdr-Familie gibt es noch die Zugriffsfunktionen First, Second, Tird, Fourth, Fifth, Sixth, Seventh, Eighth, Ninth und Tenth, die eine Liste als Argument erhalten und das erste, zweite, dritte, ..., zehnte Element der Liste als Ergebnis liefern. Die Funktion Nth verlangt eine nichtnegative ganze Zahl n und eine Liste als Argument. Nth liefert das n-te Element der Liste, wobei dem Car der Liste das nullte Element mit Nth entspricht.

Beispiele:
(nth 4 '(a b c d e f)) ⇒ E
(nth 0 '(a b c d)) ⇒ A

Die mehrmalige Anwendung der Funktion Cdr auf eine Liste wird durch Nthcdr erreicht. Wie Nth erwartet Nthcdr eine nichtnegative ganze Zahl und eine Liste als Argumente:

(nthcdr 4 '(a b c d e f))
⇒ (E F)
(nthcdr 28 '(L I S T E))
⇒ NIL

Durch Verschachtelung von mehreren Cons-Aufrufen können entsprechend lange Listen aufgebaut werden. Solche Listen lassen sich aber einfacher durch die Funktionen List bzw. List* erzeugen. Die Funktion List akzeptiert beliebig viele Argumente, die zu einer Liste zusammengefaßt werden.

(list) ⇒ NIL
(list 1 2 3)
== (cons 1 (cons 2 (cons 3 nil)))
⇒ (1 2 3)
(list (list 'a 1) (list 'b 2))
⇒ ((A 1) (B 2))

Die Funktion List* benötigt mindestens ein Argument, wobei das letzte Argument eine Liste sein muß. Als Ergebnis wird eine Liste konstruiert, die aus den ersten (n-1) Argumenten und den Elementen der Liste des letzten Arguments besteht.

(list* 1 2 '(3 4 5))
== (cons 1 (cons 2 '(3 4 5)))
⇒ (1 2 3 4 5)
(list* nil) ⇒ nil
(list* 'a 'b 'c nil) ⇒ (A B C)

Die Funktion Append ist ebenfalls eine Funktion zur Konstruktion von Listen. Sie erwartet zwei Listen als Argumente und bildet aus den Elementen dieser Listen eine neue Liste.

(append '(a b c) '(d e f))
⇒ (A B C D E F)
(append nil '((b) (c))) ⇒ ((B) (C))
(append '(radius) 2.7) nil)
⇒ (RADIUS 2.7)

Die einstellige Funktion Length liefert die Länge einer Liste, das heißt die Anzahl der Listenelemente des obersten Listenniveaus.

(length '(a b)) ⇒ 2
(length '((a b c d e f)) ⇒ 1
(length nil) ⇒ 0

Mit der Funktion Reverse können die Listenelemente auf dem obersten Niveau in ihrer Reihenfolge vertauscht werden:

(reverse '(a b c)) ⇒ (C B A)
(reverse '((a b) (c d)) ⇒ ((C D) (A B))
(reverse nil) ⇒ nil

Sowohl Length als auch Reverse betrachten das ihnen übergebene Argument als eine Liste von Elementen, unabhängig davon, ob die Elemente wiederum Listen oder Atome sind.

Arithmetische Funktionen

Neben den Funktionen für die Grundrechenarten stehen in Lisp traszendente Funktionen, Funktionen für logische Operationen mit ganzen Zahlen, für Operationen mit Teilen ganzer Zahlen (Bytes) und für die Erzeugung von Zufallszahlen bereit.

Die Grundfunktionen +, −, ∗ und / sowie die Funktionen Max (Maximum von n Zahlen), Min (Minimum von n Zahlen), Gcd (größter gemeinsamer Teiler) und Lcm (kleinstes gemeinsames Vielfaches) können beliebig viele Argumente erhalten. Alle arithmetischen Funktionen arbeiten für sämtliche Zahlentypen, man sagt, sie sind *generisch*. Beispiele für arithmetische Funktionsaufrufe:

(+ 2 3 5.5) ⇒ 10.5
(− −23.8) ⇒ 23.8
(∗ 9 3.0) ⇒ 27.0
(/ 27 3) ⇒ 9
(max 2 4 7 3) ⇒ 7
(min 2 4 7 3) ⇒ 2
(gcd 63 −42 35) ⇒ 7
(lcm 14 35) ⇒ 70
(mod 13 4) ⇒ 1 ; 13 modula 4
(ganzzahliger Rest der Division)
(abs −26.5) ⇒ 26.5 ;
der absolute Betrag von −26.5
(abs 5) ⇒ 5
(expt 2 3) ⇒ 8 ; 2^3
(sqrt 4.0) ⇒ 2.0 ; Quadratwurzel
(random 10) ⇒ 7 ; Zufallszahl
(Random n) liefert eine Zufallszahl, die größer oder gleich Null, aber kleiner als n ist. Der arithmetische Ausdruck

$$\frac{-7+\sqrt{7^2-4\cdot 2\cdot 5}}{2\cdot 2}$$

sieht in der Präfix-Notation von Lisp so aus:
(/ (+ −7 (sqrt (− (expt 7 2) (∗ 4 2 5)))) (∗ 2 2))

Übung 3

a) Berechnen Sie folgende Ausdrücke mit Lisp:
2 − 4 + 6 + 8
1 + 15 − 7.0 + 23 − 10.0 − 22
123456789 + 987654321
3.8 + 7.2 − 23.75 + 53.219

b) Erproben Sie die Fehlerreaktion von Lisp, wenn eine Funktion mit zu vielen, mit zu wenigen oder mit falschen Argumenten aufgerufen wird. Versuchen Sie es zum Beispiel mit (car nil '(eine Liste)), (cons 23)) und (+ 34 '(a b c)).

Die Definition von Funktionen

Bestimmte Symbole (z. B. + oder Cons) können auch als Funktionen verwendet werden. Diese Eigenschaft resultiert aus der Zuordnung zu einer Vorschrift, die dann abgearbeitet wird, wenn das Symbol in *funktionaler Stellung* (als erstes Element in einer Liste) auftritt. Diese Eigenschaft wird auch als die funktionale Eigenschaft des Symbols bezeichnet. Der Nutzer kann mit Hilfe der Form **(defun** <Name> <Parameter> <Körper>**)** selbst Funktionen definieren, kann also dem durch <Name> angegebenen Symbol eine funktionale Eigenschaft zuordnen, die durch die Parameter und den Körper spezifiziert wird. Defun ist eine Spezialform, die ihre Argumente nicht auswertet. Das Ergebnis von Defun ist der Name der definierten Funktion. Die Parameter werden durch eine Liste von Symbolen beschrieben, die die formalen Argumente der Funktion darstellen. Beim Aufruf der selbstdefinierten Funktion werden die aktuellen Argumente ausgewertet und der Reihe nach an die Symbole der Parameterliste gebunden. Durch Notation dieser Symbole innerhalb des Körpers kann auf diese Bindung Bezug genommen werden.

(defun zweites (L)
; Ermitteln des zweiten Elementes einer Liste
(car (cdr L)))

Beim Aufruf von (Zweites '(A B C D)) wird nach Auswertung des Arguments die Liste (A B C D) an das Symbol L gebunden. Mit dieser Bindung wird anschließend der Funktionskörper (Car (Cdr L)) abgearbeitet, wobei die Auswertung von L das Ergebnis (A B C D) liefert. Nach Abarbeitung des Funktionskörpers wird die Bindung wieder rückgängig gemacht und das Ergebnis B als Gesamtergebnis des Funktionsaufrufes zurückgegeben.

Übung 4

4a) Definieren Sie je eine Funktion zum Umrechnen von Grad Fahrenheit in Grad Celsius und umgekehrt nach folgenden Formeln:
C = (F+40)/1.8−40 und
F = (C+40)∗1.8−40.

4b) Definieren Sie eine Funktion Swap, die als Argument eine zweielementige Liste erhält und diese umdreht. Verwenden Sie dazu nur Car-Cdr-Funktionen und Cons. *Beispiel:*
(swap '(a b)) ⇒ (b a), nicht (b . a)!

4c) Überwachen Sie das Input-Output-Verhalten der Funktion Swap durch Aufruf der Form (trace swap). Rufen Sie anschließend mehrmals Swap auf. Trace ist eine Spezialform, der mehrere Funktionsnamen übergeben werden können. Mit (untrace swap) oder (clear swap) kann das Überwachen wieder ausgeschaltet werden.

Aufgabe 1

Definieren Sie eine Funktion Rotate-left, die eine beliebig lange Liste als Argument erhält. Das Ergebnis soll eine neue Liste sein, bei der das erste Argument der Ausgangsliste das letzte Element der Ergebnisliste ist. Beispiele:
(rotate-left ' (a b c d e f g))
⇒ (b c d e f g a)
(rotate-left (rotate-left '(a b c)))
⇒ (c a b)

Aufgabe 2

Definieren Sie zu der Funktion Rotate-left die Umkehrfunktion Rotate-right.

Prädikate und Konditionale

Um kompliziertere Funktionen schreiben zu können, werden Prädikate benötigt. Prädikate liefern als Ergebnis immer die Aussage *wahr* oder *falsch*. In Lisp wird für *falsch* das Symbol NIL verwendet, und jeder Wert ungleich NIL bedeutet *wahr*, also auch das Symbol T, das im Normalfall dafür benutzt wird.

Typprädikate

Typprädikate sind die wichtigsten und auch die am häufigsten verwendeten Prädikate in Lisp. Sie geben Auskunft darüber, ob ein Objekt von einem bestimmten Typ ist. Wir kennen bis jetzt Atome, Zahlen, Symbole, Zeichenketten und Listen. Um nichtleere Listen von Atomen unterscheiden zu können, gibt es den Datentyp Cons. Das Symbol NIL ist das einzige Element des Datentyps Null, der damit nicht nur Teiltyp von Symbol, sondern auch von List ist. Bild 1 zeigt, wie unser Typbaum bis jetzt aussieht.

Die jeweiligen Typprädikate heißen Atom, Numberp, Fixnump, Floatp, Stringp, Symbolp, Listp, Null und Consp. Ihre Wirkungsweise erkennt man am besten an den folgenden Beispielen:

(atom nil) ⇒ T
(atom 'a) ⇒ T
(atom '(a b c)) ⇒ NIL
(numberp 33) ⇒ T
(numberp 27.86) ⇒ T
(numberp 'five) ⇒ NIL
(fixnump 24) ⇒ T
(floatp 24) ⇒ NIL
(floatp 24.0) ⇒ T
(stringp "22.3") ⇒ T
(stringp 'string) ⇒ NIL
(listp '(a b c)) ⇒ T
(listp 'symbol) ⇒ NIL
(listp nil) == (listp ()) ⇒ T
(consp '(1 2 3)) ⇒ T
(consp NIL) ⇒ NIL
(null nil) ⇒ T

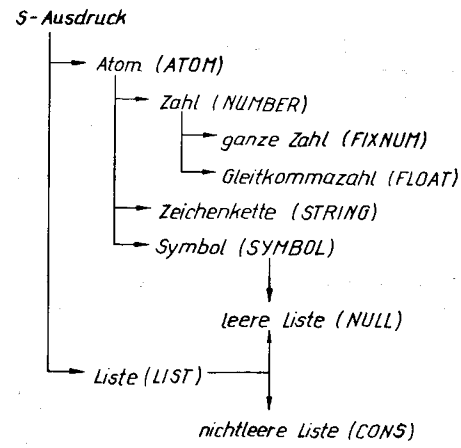


Bild 1 Die Typhierarchie in Lisp

Alle Typprädikate lassen sich auch durch das Prädikat Typep realisieren:
(typep <Objekt> <Typ>**)**
liefert T, wenn das Argument <Objekt> vom Typ <Typ> ist, ansonsten NIL.

(typep 3 'number) ⇒ T
(typep 3 'fixnum) ⇒ T
(typep 3 'float) ⇒ NIL

Vergleichsprädikate

In Lisp gibt es verschiedene Gleichheitsbegriffe und daher auch verschiedene Prädikate zum Testen auf Gleichheit. Der einfachste Test ist der Test auf physische Gleichheit (Eq). Eq liefert T, wenn die beiden Argumente physisch gleich sind:
(eq 'g 'g) ⇒ T
(eq '(a b c) '(a b c)) ⇒ NIL
Will man für zwei unterschiedliche Listen prüfen, ob sie die gleichen Elemente besitzen, so muß man Equal verwenden.
(equal '(1 2 3) '(1 2 3)) ⇒ T, aber
(eq '(1 2 3) '(1 2 3) ⇒ NIL,
denn bei jedem Einlesen von (1 2 3) wird physisch eine neue Liste erzeugt.
Zahlen werden mit den Funktionen =, >, <, <= und >= verglichen, wobei diese beliebig viele Argumente akzeptieren. Beispielsweise liefert das Prädikat <= als Ergebnis *wahr*, wenn die Folge der übergebenen Argumente eine monoton steigende Zahlenfolge ist. Mit

diesen Funktionen lassen sich sehr einfach Intervalltests formulieren:

(>= 9 8 7 7 5 4 2) ⇒ T
(> 9 8 7 7 5 4 2) ⇒ NIL
(= 3 3.0) ⇒ T, während aber gilt
(equal 3 3.0) ⇒ NIL.

Zum Vergleich von Zeichenketten werden die Funktionen *String=*, *String>*, *String<*, *String<=*, *String>=* und *String/=* verwendet. Diese zweistelligen Funktionen können auch auf Symbole angewendet werden:

(string< "abc" "abd") ⇒ T
(string= "foo" "FOO" ⇒ NIL
(string= "LISP" 'lisp) ⇒ T

Der letzte Vergleich liefert T, weil Lisp alle Buchstaben in Symbolen zu Großbuchstaben umwandelt. Die Funktionen String-equal, String-greaterp, String-lessp, String-not-greaterp, String-not-lessp und String-not-equal ignorieren beim Vergleich die Unterschiede zwischen Groß- und Kleinbuchstaben.

Logische Verknüpfungen

Mehrere Prädikate lassen sich aussagenlogisch durch
(and <wert-1> ... <wert-n>),
(or <wert-1> ... <wert-n>) *und*
(not <wert>)
verknüpfen. Die Funktion *Not* und die Spezialformen And und Or arbeiten so, wie man es von der Aussagenlogik her gewohnt ist. And wertet seine Argumente nur soweit aus, bis es auf eines trifft, das NIL liefert; die restlichen Argumente werden nicht beachtet. Or hingegen wertet nur solange aus, bis es auf ein Argument trifft, das einen Wert ungleich NIL liefert. Zum Beispiel werden bei
(and t t nil) ⇒ NIL
nacheinander alle Argumente ausgewertet, wobei
(or t t nil) ⇒ T
nur das erste Argument auswertet. Die Funktion *Not* arbeitet wie Null und liefert als Ergebnis T, wenn sie als Argument NIL erhielt.

```
(defun constantp (x)
    (or (null x) (eq x t)
      (numberp x) (stringp x)
      (and (consp x)
        (eq (car x) 'quote))))
```

Programmverzweigungen

Die Spezialform If stellt die einfachste Möglichkeit dar, Programmverzweigungen zu realisieren. If ist mit den If-then-else-Konstruktionen algebraischer Programmiersprachen vergleichbar und hat folgende Syntax:
(if <test> **<then>** [**<else>**])
If wertet zuerst die Form *Test* aus. Ist das Ergebnis ungleich NIL, wird die *Then*-Form ausgewählt, sonst die *Else*-Form. Diese ausgewählte Form wird anschließend ausgewertet und liefert das Gesamtergebnis des If-Aufrufes. Wurde die Else-Form weggelassen, so wird dafür NIL genommen.

(if (numberp 3829) 'ja 'nein) ⇒ JA
(if (consp 'a) 'cons 'kein-cons)
⇒ KEIN-CONS
(if (listp 392) 'liste) = =
(if (listp 392) 'liste nil) ⇒ NIL

Da eine Verschachtelung vieler If-Formen schwer lesbar ist, gibt es die Spezialform Cond – ähnlich den Swith- oder Case-Anweisungen anderer Programmiersprachen – mit der folgenden Syntax:
(cond (<prädikatform-1> <form-11>
... <form-1m>**)**
    (<prädikatform-n> <form-n1>
... <form-nk>**))**
Die erste Prädikatform wird ausgewertet. Liefert sie einen Wert ungleich NIL, so werden alle nachfolgenden Formen <form-11> bis <form-1m> ausgewertet und die Abarbeitung von Cond wird abgebrochen, ansonsten wird zur zweiten Prädikatform übergegangen usw. Als letzte Prädikatform empfiehlt es sich, T anzugeben. Da dies immer ungleich NIL ist, wird die Abarbeitung der letzten Alternative gesichert, wenn alle anderen verworfen werden mußten. Als Beispiel dafür wird die Funktion Typep definiert:

```
(defun typep (objekt typ)
    (cond ((eq typ 'cons) (consp objekt))
      ((eq typ 'list) (listp objekt))
      ((eq typ 'symbol) (symbolp objekt))
      ((eq typ 'atom) (atom objekt))
      (t "unbekannter Datentyp")))
```

Aufgabe 3

Definieren Sie eine Funktion SIMP+, die einen arithmetischen Ausdruck vereinfacht, wobei nur die Regeln der Addition berücksichtigt werden (z. B. 0 + n = n). Als Argument erhält die Funktion eine Liste, die aus dem Element +, dem linken Operanden und dem rechten Operanden besteht. Als Operanden können Zahlen und Symbole verwendet werden.

(simp+ '(+ 27.4 63.2)) ⇒ 100.6
(simp+ '(+ 0 b)) ⇒ B
(simp+ '(+ a c)) ⇒ (+ a c)

Im letzten Beispiel ist keine Vereinfachung möglich.

Literatur

Charniak, E.; Riesbeck, C. K.; McDermott, D. V.; Meehan, J. R.: Artifical Intelligence Programming, Lawrence Erlbaum Associates, Hillsdate, New Jersy 1987

Steele, G. L.: Common LISP: The Language, Digital Press 1984.

Stoyan, H.: Programmiermethoden der Künstlichen Intelligenz. Band 1, Studienreihe Informatik, Springer-Verlag, Berlin, Heidelberg, New York, London, Paris, Tokio 1988

Winston P. H., Horn B. K. P.: LISP, Addison-Wesley Publishing Company, Reading (Mass.) 1988

✉ KONTAKT ☎

Akademie der Wissenschaften der DDR, Zentralinstitut für Kybernetik und Informationsprozesse, Kurstraße 33, Berlin, 1086; Tel. 2 03 76


```
1)  (DEFUN ROTATE-LEFT (LISTE)
           (APPEND (CDR LISTE)
                   (LIST (CAR LISTE))))

2a) (DEFUN ROTATE-RIGHT1 (LISTE)
           (REVERSE
              (ROTATE-LEFT (REVERSE LISTE))))

2b) (DEFUN ROTATE-RIGHT2 (LISTE)
           (APPEND
              (NTHCDR (- (LENGTH LISTE) 1)
                      LISTE)
              (REVERSE
                 (CDR (REVERSE LISTE)))))

3)  (DEFUN SIMP+ (FORMEL)
       (COND ((AND (NUMBERP (SECOND FORMEL))
                   (NUMBERP (THIRD FORMEL)))
              (+ (SECOND FORMEL)
                 (THIRD FORMEL)))
             ((AND (NUMBERP (SECOND FORMEL))
                   (ZEROP (SECOND FORMEL)))
              (THIRD FORMEL))
             ((AND (NUMBERP (THIRD FORMEL))
                   (ZEROP (THIRD FORMEL)))
              (SECOND FORMEL))
             ((EQ (SECOND FORMEL)
                  (THIRD FORMEL))
              (LIST '* 2 (SECOND FORMEL)))
             (T FORMEL)))
```

Bild 2 Lösungen der Aufgaben 1 bis 3

Zur Lösung der Aufgabe 3 sei folgendes angemerkt: Die Funktion besteht aus einem Cond, in dessen Zweigen die Bedingungen für eventuelle Vereinfachungen geprüft werden. Der erste Zweig sorgt für die Ausführung der Addition, wenn beide Summanden Konstanten sind. Der zweite und der dritte Zweig sorgen dafür, daß keine unnötige Addition mit 0 durchgeführt wird. Der vierte Zweig wandelt die Addition zweier gleicher Summanden in eine Multiplikation um. Der fünfte Zweig hat als Bedingung T, das heißt, in jedem anderen Fall ist keine Vereinfachung möglich.

TERMINE

Forth-Lehrgänge

WER? Kammer der Technik Suhl
WANN?
1. Lehrgang: 3. bis 7. 9. 1990
2. Lehrgang: 10. bis 14. 9. 1990

WO? Suhl
WAS?
- Möglichkeiten von Forth-83
- Programmieren selbständig ausgewählter Probleme
- Überblick über verschiedene Forth-Prozessoren

WIE Auskünfte und Anmeldungen bitte bei Kammer der Technik Suhl, Bereich Weiterbildung, Postfach 510, Suhl, 6000; Tel. 2 21 12 (Kolln. Keller)

Dr. Finsterbusch

1. Problemseminar „Programmierung von 32-Bit-Minicomputersystemen"

WER? Kammer der Technik, Fachverband Elektrotechnik, Fachausschuß Minicomputersysteme und DECUS München e. V.
WANN? 17. bis 21. September 1990
WO? Cottbus
WAS?
- Betriebssysteme und Standardsoftware für Minicomputersysteme wie VAX-11, K 1840, K 1820, CM 52/12, Elektronika-82, CM 1700
- Generierung, Management, Optimierung, Anwendungsprobleme und Anwendungserfahrungen

WIE? Teilnahmemeldungen richten Sie bitte an: Kammer der Technik, Bezirksvorstand Cottbus, Stadtpromenade 3, Postschließfach 90/1 Cottbus, 7500; Tel. 2 40 95

Prof. Dr. Horn

Fortsetzung von Seite 206

FloatExs 8087-Exception-Handling
ProcTrace Aufzeichnen von Prozeduraufrufen von Variablenwerten

Extra geliefert wird ein sogenanntes TechKit, das die Assembler-Quellen der Run-Time-Bibliothek und den TopSpeed-Assembler enthält. Seine Mnemonik weicht aber vom Microsoft-quasi-Standard ab. Er ist einfach gehalten, unterstützt keine Macros etc. Daß aber Kommentare in Modula-Notation angegeben werden müssen – (* und *) –, das Semikolon dafür aber zur Trennung mehrerer Befehle auf einer Zeile vergeben ist, läßt die Übernahme von Assemblerquellen kaum zu.

```
Import analysis of sieve
------------------------

sieve          IMPORTS IO, Sieve ;
IO             IMPORTS Lib, SYSTEM, AsmLib, Str, IO ;
Sieve          IMPORTS IO, Sieve ;
Lib            IMPORTS SYSTEM, AsmLib, Lib ;
SYSTEM
AsmLib
Str            IMPORTS MATHLIB, Lib, SYSTEM, AsmLib, Str ;
MATHLIB

Date/Time Anomaly Check
-----------------------

No date/time anomalies detected
```

Ein nützliches Tool ist das Programm ANALYZE, das die Abhängigkeiten und die Aktualität der einzelnen Module zeigt (Bild oben). Weiterhin im TechKit enthalten sind die Module zur Unterstützung residenter Programme, der seriellen Schnittstelle und von Expanded Memory.
Ein neuer Gedanke ist die sogenannte TechLine. Damit können Sie aus der Entwicklungsumgebung heraus über ein Modem die Entwickler in London direkt an die Leitung holen...

Uwe Schulze

Literatur

/1/ Schulze, U.: Im Test: Compiler für MS-DOS. Mikroprozessortechnik, Berlin 4 (1990) 4, S. 103
/2/ TopSpeed Modula-2 TechKid. Jensen & Partners International 1988
/3/ Die TopSpeed-Story. Toolbox (1989) 6, S. 8
/4/ Braun, M.; Guther, J.: Modula-2 wird erwachsen. mc, München (1988) 9 S. 68
/5/ Hauptmann, S.: Modula-2: Pascal ohne die Nachteile von Pascal. Mikroprozessortechnik, Berlin 2 (1988) 10, S. 296
/6/ JPI TopSpeed Modula-2 Benutzerhandbuch, Jensen & Partners International and ALISYS 1988

Wegbereiter der Informatik

VANNEVAR BUSH

*** 1890 Massachusetts**
† 1974 Washington (?)

Während in der Digitalrechentechnik historisch eindeutig zurückverfolgt werden kann, wer die 1. Rechenmaschine konstruiert hat (nämlich 1623 der Tübinger Universitätsprofessor Wilhelm Schickard), so kann in der Analogrechentechnik die Prioritätsfrage nach dem 1. Analogrechner nicht mit der gleichen Klarheit beantwortet werden. Es steht zwar fest, daß der englische Physiker Sir William Thomson (Lord Kelvin) 1876 das auf der Rückkopplung beruhende analogrechentechnische Programmierprinzip zur Lösung gewöhnlicher Differentialgleichungen angegeben und auch einen mechanischen Integrierer konzipiert hat. Aber der 1. komplette mechanische Analogrechner ist nach derzeitigem Kenntnisstand der Differentialanalysator des amerikanischen Elektro-Ingenieurs und Physikers Vannevar Bush (die zeitlichen Angaben für den Konstruktionsbeginn seiner Maschine schwanken zwischen 1931 und 1936). Ein weiterer mechanischer Differentialanalysator wurde – ebenfalls in den 30er Jahren – in Oslo von Sven Rosseland gebaut. Jedoch den ersten universell verwendbaren elektronischen Analogrechner konstruierte 1941 der Elektro-Ingenieur Helmut Hoelzer, ein Mitarbeiter Wernher von Brauns in Peenemünde; darüber wird noch gesondert zu berichten sein.
Das Ergebnis von V. Bushs Entwicklung war eine gewaltige Maschine von 100 Tonnen Gewicht. Sie enthielt 2000 Röhren, mehrere tausend Relais und 150 Motoren, die für den Rechenbetrieb der Maschine erforderlich waren. Sie war mit 18 Integrierern ausgerüstet und erlaubte demnach die Auflösung von Differentialgleichungssystemen mit maximal 18 Gleichungen erster Ordnung.
Vannevar Bush entstammt einer protestantischen Pfarrersfamilie, deren Vorfahren schon über mehrere Generationen in Massachusetts ansässig waren. Er studierte bis 1913 am *Tufts College*, in dem er das „Master of Science"-Examen ablegte. Danach arbeitete er im Prüffeld der General Electric Co. und ein Jahr später in der Inspektionsabteilung der amerikanischen Marine. Gleichzeitig hielt er am *Tufts College* mathematische Vorlesungen und wurde dort Lehrbeauftragter für Elektrotechnik. Im Jahre 1916 promovierte er an dem berühmten *Massachusetts Institute of Technology* (MIT) zum Dr. Ing. Nach dem 1. Weltkrieg erhielt er am MIT, wo ab 1919 ja auch Norbert Wiener lehrte, zunächst eine außerordentliche Professur und darauf ein Ordinariat für elektrische Energieübertragung. Im Jahre 1932 wählte man ihn zum Vizepräsidenten des MIT, und 1938 wurde er zum Präsidenten der *Carnegie Institution of Washington* – einer der maßgeblichsten Forschungsgesellschaften in den USA – ernannt; dieses Amt bekleidete er ohne Unterbrechung bis zu seiner Emeritierung (1955).
Bush gehört zu denjenigen Gelehrten, die neben ihrer Tätigkeit in Forschung und Lehre noch gewichtige beratende oder leitende Funktionen in staatlichen Ämtern innehatten. So wurde er 1940 vom Präsidenten der USA zum Vorsitzenden des *National Defense Research Committee* berufen; damit wurde er eine zentrale Persönlichkeit im damaligen amerikanischen Verteidigungsprogramm. Ab 1944 legte Bush auf Vorschlag von Präsident Roosevelt seine Erfahrungen auf dem Gebiet der militärischen Verteidigung in einer Denkschrift nieder („Science, the Endless Frontier"), die in den USA offenbar gewisse Berühmtheit erlangt hat und dem amerikanischen Kongreß in der Nachkriegszeit als Studienunterlage diente.
Nach 1936 hatte Bush auch mit der Ausarbeitung eines Entwurfs für einen elektronischen Digitalrechner begonnen, der mit einer Taktfrequenz von 10000 Hz arbeiten sollte und dessen Operationszeit für die Multiplikation zweier 6stelliger Zahlen 0,2 Sekunden betragen sollte. Bushs Ausarbeitungen konnten wohl wegen des Krieges zunächst nicht realisiert werden, wurden aber später beim Bau des ENIAC wesentlich mit verwendet.

Dr. Klaus Biener

TERMINE

7. Fachtagung „Mikroelektronik '90"

WER? Kammer der Technik, Fachverband Elektrotechnik
WANN? 12. und 13. Dezember 1990
WO? Dresden
WAS?
- Internationaler Stand der Anwendung der Mikroelektronik
- Information zur Entwicklung und Anwendung der Mikroelektronik
- Automatisierung technologischer Prozesse
- Entwurf und Prüfung mikroelektronischer Erzeugnisse

WIE? Vortragsanmeldungen (bis zum 15. August) und Teilnahmemeldungen richten Sie bitte schriftlich an:
Kammer der Technik, Präsidium, Fachverband Elektrotechnik, Clara-Zetkin-Straße 115/117, Berlin, 1086

Hoppe

7. Problemseminar „Programmierung von 16-Bit-Minicomputersystemen"

WER? Kammer der Technik, Fachverband Elektrotechnik, Fachausschuß Minicomputersysteme und DECUS München e. V.
WANN? 10. bis 12. Oktober 1990
WO? Gera
WAS? Betriebssysteme und Standardsoftware für
- Minicomputersysteme wie PDP-11, SM 14/20, SM 52/11, I 102F, K 1630
- Generierung, Management, Optimierung, Anwendungsprobleme und Anwendungserfahrungen in Verbindung mit Eintageslehrgängen am 12. Oktober zu KMS, DECnet, C, GKS, Systemgenerierung

WIE? Vortragsmeldungen richten Sie bitte bis zum 30. Juli an: Ingenieurhochschule Mittweida, ORZ, Dr. Geisler, Platz der DSF 17, Mittweida, 9250; Tel. 5 84 70
Teilnahmemeldungen bitte an: Kammer der Technik, Bezirksvorstand Gera, Humboldtstraße 13, Gera, 6500; Tel. 2 33 38, 2 33 39

Prof. Dr. Horn

Computer-Club

Packer oder Die Kompression von Daten

Berthold Biener, Erfurt

Beim Einsatz von Personalcomputern zeigt sich ein Trend zu immer umfangreicheren Programmpaketen. Reichte beim 8-Bit-Rechner meist noch eine Diskette für alle wichtigen Systemprogramme aus, wird es bei den IBM-kompatiblen oft schon auf den 20- oder 40-MByte-Festplatten ziemlich eng. Beim System PS/2 gehen die Software-Häuser noch sorgloser mit dem Speicherplatz um, damit ihre Kunden ein Maximum an Funktionsvielfalt bekommen.

Die Anwenderprogramme erzeugen immer größere Dateien in Form von Texten, Datenbanken und Grafiken. Diese werden meist für längere Zeiten archiviert, selbst wenn sie nicht mehr im ständigen Gebrauch sind, oder sie sollen mit einem Datenfernübertragungssystem (z. B. Modem) über das Telefonnetz auf mehr oder weniger weit entfernte PCs übertragen werden. Hier ergibt sich jedoch eine Möglichkeit, sowohl die Speichermedien effektiver zu nutzen, als auch die Übertragungszeiten bei gleicher Übertragungsrate zu verkürzen, um damit am Ende Gebühren zu sparen.
Die Informationsdichte der meisten Dateien ist ziemlich gering. In Quellprogrammen häufen sich am Zeilenanfang Leerzeichen oder Tabulatoren; Textdateien nutzen sowieso nicht gleichmäßig alle mit 8 Bit darstellbaren 256 Kodes. Auch in Datenbanken bleibt meist ein erheblicher Teil des Speicherplatzes ungenutzt, und in Grafiken gibt es große Flächen, auf denen sich *nichts ändert*. An dieser Stelle setzen Methoden zur Erhöhung der Informationsdichte durch die Verkürzung von Dateien ein.
Derartige Methoden werden in sogenannten Packern angewendet. Packer sind relativ komplexe Programme, die, meist als *Public Domain Software* kostenlos oder *Shareware* gegen einen geringen Unkostenbeitrag angeboten, das Verdichten von einzelnen Dateien, Dateigruppen oder ganzen Verzeichnissen ermöglichen. Um die Originaldateien wiederherzustellen, wird ein entsprechender *Entpacker* benötigt. Bei neueren Produkten sind Packer und Entpacker in einem Programm zusammengefaßt. Diese Programme beherrschen verschiedene Verdichtungsalgorithmen, aus denen sie den optimalen Algorithmus selbsttätig auswählen, und sind weitgehend zueinander kompatibel. Darüber hinaus ist es mit einigen Programmen möglich, sich selbst *auspackende* Dateien zu erzeugen, die auch aus mehreren Dateien bestehen können.
In der DDR am weitesten verbreitet sind der Packer PKARC und der Entpacker PKXARC von Phil Katz (USA). Weiterentwicklungen davon sind PKPAK und PKZIP, die jeweils beide Verarbeitungsrichtungen beherrschen. Diese Produkte erreichen je nach Dateityp Verkürzungen auf 40 bis 70 Prozent der ursprünglichen Länge, wobei die komprimierten Dateien in sogenannte Archivdateien abgespeichert werden. Beim eigentlichen Packen wird meist ein Algorithmus angewendet, der in einem iterativen Verfahren schrittweise versucht, jeweils zwei Datenbytes durch einen 1-Byte-Zeiger auf die Stelle des vorherigen Auftretens dieser Kombination zu ersetzen.
Zwei andere Kompressionsverfahren werden im folgenden vorgestellt (Bild 1). Sie sind zwar relativ einfach zu programmieren, daher aber auch nicht sehr leistungsfähig.
Das einfachste Verfahren ist das ‚Packen' im engeren Sinne.
Hierbei erfolgt eine Verdichtung, wenn in einer Datei mehrmals hintereinander die gleichen Datenkodes auftreten. In diesem Fall wird ein Signalbyte (z. B. $FF) ausgegeben, dahinter die Anzahl der Wiederholungen und der Datenkode selbst. Tritt der als Signalbyte verwendete Kode in der Datei auf, so wird die Wiederholungsanzahl mit Null angegeben. Die Verdichtung ist nur sinnvoll, wenn der gleiche Datenkode mehr als dreimal ununterbrochen hintereinander folgt. Die Dekompression ist sehr einfach realisierbar. Wird das Signalbyte erkannt, wird bei folgender Null dieses in die Zieldatei geschrieben, andernfalls stellt das dem Signalbyte folgende Byte die Anzahl dar, wie oft das als Drittes stehende Datenbyte in die Zieldatei geschrieben werden muß. Dieses Verfahren kann für alle Dateitypen verwendet werden, bringt aber nur begrenzte Vorteile.
Für Textdateien bietet sich ein anderes Verfahren an. Dieses geht von der unterschiedlichen Häufigkeit des Auftretens der möglichen (ASCII-) Zeichen aus. Die geordnete Verteilung der Zeichen einer Textdatei weist am häufigsten Leerzeichen, Zeilenwechsel sowie die Kleinbuchstaben e, i, r, n, t, a auf. Das Kompressionsverfahren beruht darauf, daß die 12 häufigsten Zeichen durch eine 4-Bit-Darstellung ersetzt werden. Damit werden über 60 Prozent der im Text auftretenden Zeichen erfaßt. Weitere 32 Zeichen werden als 8-Bit-Darstellung verarbeitet, während der Rest der Zeichen – meist Buchstaben wie q, x, y, Großbuchstaben und Sonderzeichen – durch einen Signalcode von 4 Bit auf 12 Bit verlängert werden. Da diese Zeichen insgesamt aber nur 10 bis 20 Prozent des Textes ausmachen, beträgt die Verlängerung 5 bis 10 Prozent, so daß im Endeffekt mit einer Verkürzung von Textdateien auf 70 bis 80 Prozent zu rechnen ist. Beim Test mit realen Dateien ergaben sich Verkürzungen auf rund 70 Prozent.
Zunächst muß die gesamte Datei analysiert werden, um die häufigsten Zeichen herauszufinden. Es wird einfach die Anzahl des Auftretens jedes Zeichens gezählt und diese Tabelle sortiert. Die Sortierung kann nach 44 Elementen abgebrochen werden. Diese werden in einen Vorblock der Zieldatei abgespeichert. Sollte sich herausstellen, daß die Anzahl der ersten 12 Elemente kleiner ist als die Anzahl der nicht mehr sortierten Elemente, kann der Kompressionsversuch abgebrochen werden, da keine Einsparung von Speicherplatz erzielbar ist. Dies tritt bei Binärdateien (z. B. Maschinenprogrammen) auf.
Nach erfolgter Analyse wird die Quelldatei auf den Anfang zurückgesetzt und es beginnt die eigentliche Kompression. Nach dem Einlesen eines Zeichens wird überprüft, ob sich dieses Zeichen mindestens noch zweimal wiederholt, wenn das der Fall ist, wird ein Packen ermöglicht, indem das Signalnibble (Halbbyte) $E und die Wiederholungsanzahl (maximal 15=$F) ausgegeben wird. In jedem Fall wird nun die Tabelle nach dem zu bearbeitenden Datenbyte durchsucht. Wird es zwischen der ersten und der zwölften Tabellenposition gefunden, ist der Tabellenoffset die auszugebende 4-Bit-Darstellung. Liegt der Tabellenoffset zwischen 12 und 43, wird zu diesem Wert $B4 addiert und dieser als 8-Bit-Darstellung ausgegeben. Diese Kodes sind somit an den Signalnibbles $C und $D zu erkennen. Wird das Datenbyte überhaupt nicht in der Tabelle gefunden, muß es unverändert hinter dem Signalnibble $F ausgegeben werden. Auf diese Weise erfolgt die Bearbeitung bis zum Ende der Quelldatei. Als Endekennung in der Zieldatei werden nun noch drei Bytes Null ausgegeben.
Die Wiederherstellung des ursprünglichen Textes setzt die Kenntnis der Umkodiertabelle voraus, die aus diesem Grund als erstes eingelesen werden muß. Danach beginnt die Bearbeitung der Daten. Dazu wird das jeweils nächste Nibble eingelesen. Handelt es sich dabei um das Signalnibble $E, folgen darauf die Anzahl und das erste (oder einzige) Nibble des zu rekonstruierenden Datenbytes. Wurde das Signalnibble $E nicht erkannt, ist die Anzahl 1. Nun wird kontrolliert, ob ein anderes Signalnibble vorliegt.
Die Signalnibbles $C und $D werden als High-Teil und das folgende Nibble als Low-Teil eines Bytes verwendet, das nach Abzug von $B4 den Tabellenoffset ergibt. Nach dem Signalnibble $F wird das Datenbyte in 2 Nibbles direkt eingelesen. Ist das eingelesene Nibble kein Signalnibble, ist es

```
program COMPRESS;
{ Demo-Programm zur Kompression und Dekompression von Dateien }
{ Aufruf:  COMPRESS filename.typ [/B]  [/D] }
{        (Binaerdatei, $1A nicht Ende)  (Quelldatei loeschen) }

const cpt:array[0..2] of char='CPT';
      msiz=16384;
type fi=string[14];
var i,j,k,l,m,n,p,po,pc,err:integer;
    asc,loe,e,ef,eg:boolean;
    fn,fo:fi;
    fb:array[0..14] of byte absolute fo;
    dv,tn,tyo:string[4];
    tyc:array[0..4] of char absolute tyo;
    vert:array[0..255] of real;
    code:array[0..255] of byte;
    rbuf:array[0..255] of byte;
    wbuf:array[0..16511] of byte;
    f1,f2:file;

function UPPER(s:fi):fi;
{ Wandelt String in Grossbuchstaben }
var i:integer;
    b:array[0..14] of byte absolute s;
begin
 for i:=1 to b[0] do if b[i]>$60 then b[i]:=b[i] and $5f;
 upper:=s;
end;

function SUM(i,j:integer):real;
{ Bildet die Summe von Real-Zahlen in vert }
var s:real;
    k:integer;
begin
 s:=0.0;
 for k:=i to j do s:=s+vert[k];
 sum:=s;
end;

procedure ANALYSE;
{ Bildet die Verteilung der Codes der Datei }
begin
 fillchar(vert[0],1536,0);ef:=false;
 while not eof(f1) do
  begin
   blockread(f1,rbuf,1,err);
   for j:=0 to 127 do
    begin
     k:=rbuf[j];
     vert[k]:=vert[k]+1;
     if asc and (k=$1a) then exit;
    end;
  end;
end;

procedure SORT;
{ Sortiert die ersten 44 Elemente der Verteilung }
var r:real;
begin
 for j:=0 to 255 do code[j]:=j;
 for j:=0 to 43 do
  for i:=j+1 to 255 do
   if vert[i]>vert[j] then
    begin
     r:=vert[j];vert[j]:=vert[i];vert[i]:=r;
     k:=code[j];code[j]:=code[i];code[i]:=k;
    end;
end;
```

```
function EFFECT:boolean;
{ Kontrolliert, ob die Kompression effektiv ist }
begin
 EFFECT:=SUM(0,11)>SUM(44,255);
end;

procedure COMP;
{ Fuehrt die eigentliche Kompression aus }
begin
 blockread(f1,rbuf[128],1,err);
 i:=128;j:=64;e:=true;ef:=false;eg:=false;
 repeat
  if i>127 then
   begin
    { Neuen Block lesen }
    move(rbuf[128],rbuf[0],128);i:=i-128;eg:=ef;
    if not eof(f1)
     then blockread(f1,rbuf[128],1,err)
     else ef:=true;
   end;
  if not eg then
   begin
    L:=0;
    m:=rbuf[i];
    eg:=asc and (m=$1a);
    { Test auf Wiederholung }
    repeat L:=L+1 until (rbuf[i+L]<>m) or (L>=15);
   if L>2 then
    begin
     i:=i+L;
     if e then begin wbuf[j]:=(L shl 4)+14;j:=j+1 end
          else begin wbuf[j]:=n+$E0;j:=j+1;n:=L end;
    end
   else i:=i+1;
   { Code suchen in Tabelle }
   L:=0;while (L<44) and (code[L]<>m) do L:=L+1;
   if L<44 then
    begin
     if L<12 then
      begin
       if e then n:=L
       else begin wbuf[j]:=n+(L shl 4);j:=j+1 end;
      end
     else
      begin
       L:=L+$B4;
       if e then begin wbuf[j]:=(L shr 4)+(L shl 4);j:=j+1 end
        else begin wbuf[j]:=n+(L and $f0);j:=j+1;n:=L and $0f; end;
       e:=not e
      end;
    end
   else
    begin
     { Code nicht in Tabelle }
     if e then begin wbuf[j]:=15+(m shl 4);j:=j+1;n:=m shr 4; end
          else begin wbuf[j]:=n+$F0;j:=j+1;wbuf[j]:=m;j:=j+1 end;
    end;
    e:=not e;
    if j>=msiz then
     begin
      { Pufferinhalt ausgeben }
      blockwrite(f2,wbuf[0],msiz div 128);
      move(wbuf[msiz],wbuf[0],j-msiz);
      j:=j-msiz;
     end;
   end;
 until eg;

 { Endebehandlung }
 if not e then
  begin
   wbuf[j]:=n;
   j:=j+1
  end;
 wbuf[j]:=0;wbuf[j+1]:=0;wbuf[j+2]:=0;j:=j+2;
 blockwrite(f2,wbuf[0],1+(j div 128));
end;

procedure EXPAND;
begin
 move(rbuf[144],code[0],44);
 j:=192;e:=true;k:=0;
  repeat
   if j>127 then
    begin
     { Neuen Block einlesen }
     move(rbuf[128],rbuf[0],128);j:=j-128;
     if not eof(f1) then blockread(f1,rbuf[128],1,err);
    end;
   if e then m:=rbuf[j] and 15
    else begin m:=rbuf[j] shr 4;j:=j+1 end;
   e:=not e;
   L:=1;
   if m=14 then
    begin
     { Wiederholzaehler ermitteln }
     if e then begin L:=rbuf[j] and 15;m:=rbuf[j] shr 4;j:=j+1 end
          else begin L:=rbuf[j] shr 4;j:=j+1;m:=rbuf[j] and 15 end;
    end;
   if m=15 then
    begin
     { Code nicht in Tabelle }
     if e then begin m:=rbuf[j];j:=j+1 end
      else
       begin
        m:=rbuf[j] shr 4;j:=j+1;m:=m+((rbuf[j] and 15) shl 4)
       end;
    end
   else
    { Code aus Tabelle lesen }
    if m<12 then m:=code[m]
     else
      begin
       if e then m:=16*m+(rbuf[j] and 15)
        else begin m:=16*m+(rbuf[j] shr 4);j:=j+1 end;
       e:=not e;
       m:=code[m-$B4];
      end;
    repeat
     wbuf[k]:=m;
     k:=k+1;l:=l-1;
    until l=0;
    if k>=msiz then
     begin
      { Pufferinhalt ausgeben }
      blockwrite(f2,wbuf[0],msiz div 128);
      move(wbuf[msiz],wbuf[0],k-msiz);
      k:=k-msiz;
     end;
  until (rbuf[j]=0) and (rbuf[j+1]=0);
  if k>0 then blockwrite(f2,wbuf[0],1+((k-1) div 128));
end;

{ ========== Hauptprogramm =========== }
begin
 writeln('Kompressionsprogramm (C) bidat''90');
 pc:=paramcount;
 repeat
  if pc=0 then
   begin write('Eingangsdatei:');readln(fn) end
   else fn:=paramstr(1);
  asc:=(pc<2) or not
   ((upper(paramstr(pc-1))='/B') or (upper(paramstr(pc))='/B'));
  loe:=(pc>1) and
   ((upper(paramstr(pc-1))='/D') or (upper(paramstr(pc))='/D'));
  fn:=upper(fn);
  pc:=0;if fn='$' then exit;
  p:=pos('.',fn);po:=pos(':',fn);
  if p=0 then
   begin fo:=copy(fn,po+1,length(fn)-po);tn:='' end
  else
   begin fo:=copy(fn,po+1,p-1-po);tn:=copy(fn,p+1,3) end;
   if po>0 then dv:=copy(fn,1,po) else dv:='';
  assign(f1,fn);
{$I-}
  reset(f1);
{$I+}
 until ioresult=0;
 if tn='CPR' then
  begin
   blockread(f1,rbuf[128],1,err);
   move(rbuf[128],tyc[1],3);tyc[0]:=#3;
   if tyo<>cpt then
    begin writeln(fn,' ist keine CPR-Datei !');exit end;
   move(rbuf[131],fb[0],9);
   move(rbuf[140],tyc[0],4);
   fo:=fo+'.'+tyo;
   assign(f2,fo);
   rewrite(f2);
   writeln(fo,' wird dekomprimiert.');
   EXPAND;
   close(f2);
  end
 else
  begin
   writeln(fn,' wird komprimiert.');
   ANALYSE;
   reset(f1);
   SORT;
   if EFFECT then
    begin
     fillchar(wbuf[0],64,0);
     move(cpt,wbuf[0],3);
     move(fb[0],wbuf[3],9);
     tyo:=tn;
     move(tyc[0],wbuf[12],4);
     move(code,wbuf[16],44);
     fo:=fo+'.CPR';
     assign(f2,fo);
     rewrite(f2);
     COMP;
     close(f2);
    end
   else writeln(fn,' nicht komprimierbar');
  end;
 close(f1);
 if loe then
  begin erase(f1);writeln('Eingangsdatei geloescht !'); end;
 write('Fertig !');
end.
```

Bild 1 Listing des Programms Compress.pas

direkt der Tabellenoffset. Das Datenbyte wird der Tabelle entnommen und entsprechend der Anzahl in die Zieldatei geschrieben. Die Bearbeitung endet beim Erkennen der Abschluß-Sequenz von 3 Bytes Null.

Das vorgestellte Kompressionsverfahren wurde an Dateien unterschiedlicher Typen praktisch erprobt.

Dabei lassen sich Maschinenprogramme mit diesem Verfahren nicht komprimieren, da keine auswertbare Häufigkeitsverteilung ermittelt werden kann.

Textdateien und Quellprogramme lassen sich auf etwa 70 Prozent komprimieren.

Bei Datenbanken wurde eine Verkürzung auf etwa 50 Prozent erreicht, wobei jedoch erhebliche Unterschiede möglich sind.

Grafikdateien in Form von Bitmaps lassen sich bis auf 20 Prozent verdichten.

Neue 1-Mega-Bit-Speicher von Siemens

Siemens erweitert sein Produktspektrum an 1-MBit-Speichern. Die in CMOS-Technik hergestellten DRAMs sind jetzt auch mit verringerten Zugriffszeiten von maximal 60 Nanosekunden erhältlich. Solche schnellen Speicherchips werden insbesondere für High-Speed-Anwendungen in Grafik-Systemen und in der Datentechnik benötigt.

Bei Geräten mit Batteriebetrieb ist geringe Leistungsaufnahme entscheidend. Für diesen Einsatzbereich stehen jetzt alle 1-MBit-DRAMs mit 60 bzw. 70 Nanosekunden Zugriffszeit in einer Low-power-Version zur Verfügung. Diese Bausteine benötigen im Ruhezustand nur noch 1,65 Milliwatt, ein gegenüber den Standardversionen um 80 % reduzierter Wert. *MP*

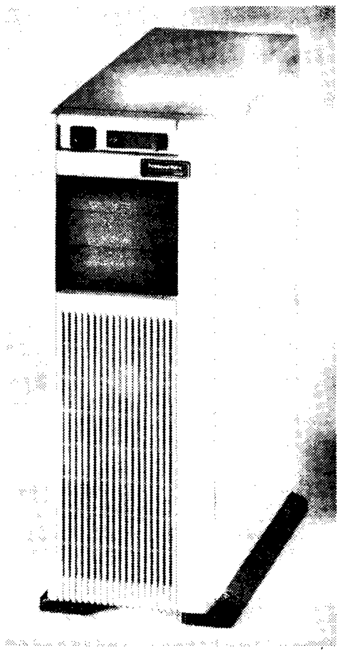
Microsoft Works 2.0 in deutsch

Das integrierte Paket Works liegt jetzt in der deutschen Version vor. Die integrierten Pakete mit den wichtigsten Anwendungen Textverarbeitung, Datenbankanwendung, Tabellenkalkulation und Datenübertragung sind für Nutzer gedacht, die nicht unbedingt alle Möglichkeiten beispielsweise der professionellen Textverarbeitung ausschöpfen wollen. Die Bedienoberfläche ähnelt dabei MS-Windows. Pull-down-Menüs und bis zu acht gleichzeitig geöffnete Fenster sind dabei nur die äußerlichen Merkmale. So ist es beispielsweise möglich, für 100 000 Wörter die Rechtschreibung zu überprüfen oder sich zu 30 000 Wörtern durchschnittlich 10 Synonyme anbieten zu lassen. Die Version 2.0 erfordert einen IBM-kompatiblen PC mit einer MS-DOS-Version ab 2.0, 512 KByte Hauptspeicher, zwei Diskettenlaufwerke mit 360 KByte oder eins mit 720 KByte/1,44 MByte und eine Festplatte. Als Grafikkarten können CGA, Hercules, MCGA oder VGA dienen. Ab sofort ist Microsoft Works 2.0 für etwa 875,– DM erhältlich, ein Update kostet etwa 175,– DM. *MP*

Neues Unix auf SPARC-Rechnern

Mit der Vorstellung ihrer neuen Abteilungsserver DRS 6000 präsentierte zur CeBIT '90 die Firma ICL als eine der ersten die Portierung der neuen Unix-Version System V Release 4.0 für RISC-Systeme. Als Entwicklungspartner von AT & T hat ICL den Referenz Source Code entwickelt, mit dessen Hilfe das neue Unix auf allen RISC-Rechnern mit der SPARC-Architektur laufen soll. An einen DRS 6000-Server können bis zu 1000 Anwendersysteme angeschlossen werden; die typischen Konfigurationen sollen allerdings zwischen 32 und 200 gleichzeitigen Nutzern liegen. Zunächst werden zwei Rechnervarianten eingeführt: Level 40 mit 17,5 MIPS und Level 50 mit 23 MIPS; die Entwicklung geht bei ICL aber weiter zu Multiprozessorsystemen mit Leistungen bis über 60 MIPS. *MP*

Neuer PostScript-RIP 4 von Linotype



Raster-Image-Prozessoren (RIP) dienen dazu, mit Rechnern erzeugte Bilder und Texte in eine Seitenbeschreibungssprache wie PostScript umzurechnen und damit Belichter oder Drucker anzusteuern.

Mit dem PostScript-RIP 3, der 1989 vorgestellt wurde, setzte Linotype neue Maßstäbe in der Verarbeitung von Halbtonbildern, die bis zu diesem Zeitpunkt besonders zeitintensiv war. Der Erfolg – eine Halbierung der Verarbeitungszeit bei hoher Auflösung – basierte auf einem neuen technischen Konzept: der Saturn-Technologie von Linotype.

Der neue PostScript-RIP 4 basiert auf dem Konzept des PostScript RIP 3, enthält jedoch standardmäßig das neue Saturn Plus-Board. So wurde die Aufbereitungszeit von Halbtonbildern im RIP bei hohen Auflösungen wiederum halbiert. Dieser Produktivitätsgewinn ist allerdings nicht der einzige Nutzen für Linotype-Kunden. Zusätzlich ergeben sich weitere Vorteile: Die Recordergeschwindigkeit der Linotronic 300 R bei 1000 Linien/cm wurde verdoppelt und die höchste Auflösung der Linotronic 500 R steigt von 666 Linien/cm auf 1000 Linien/cm. Vorhandene PostScript-RIP 3 können zu PostScript-RIP 4 aufgerüstet werden. Neue Geräte wie die Linotronic 200 SQ enthalten bereits serienmäßig einen RIP 4. *MP*

MS-DOS 4.01 jetzt auch in russisch

Am 5. April 1990 wurde in Moskau eine russische Übersetzung der Version 4.01 des weltweit am häufigsten eingesetzten PC-Betriebssystems MS-DOS von Microsoft vorgestellt. Der Gründer der Firma und Vorstandsvorsitzende der Microsoft Corp., Bill Gates, hatte es sich nicht nehmen lassen, dieses extra für die Sowjetunion produzierte Software-Paket höchstselbst zu präsentieren und damit das große Interesse der Firma am UdSSR-Markt zu dokumentieren. Als Ergebnis einer bereits seit 1988 bestehenden Zusammenarbeit zwischen Microsoft und dem sowjetisch-amerikanischen Unternehmen Dialogue entstand unter Mitwirkung von deutschen, französischen, sowjetischen und amerikanischen Spezialisten eine Version mit einer speziellen kyrillischen Bildschirmdarstellung, und es werden Drucker mit kyrillischen Fonts unterstützt. Vollständig lokalisierte Menüs, Dialogboxen und integrierte Hilfefunktionen werden ebenfalls geboten. *MP*

55 Nanosekunden bei EPROMs

EPROMs gehören seit einigen Jahren zu den langsamsten Bauelementen in einem Rechnersystem. Deshalb wird bei PCs in bestimmten Fällen der EPROM-Inhalt nach dem Einschalten in den RAM umgeladen (Shadow-RAM). Ein einfacheres Rechnerdesign erlauben aber schnellere EPROMs. Während die üblichen Zugriffszeiten von EPROMs zur Zeit noch zwischen 100 und 250 ns liegen, werden von einigen Firmen Entwicklungen wesentlich schnellerer Bauelemente gemeldet. Dazu gehört auch die International CMOS Technology Incorporation (ICT). Ihr neuer 1-MBit-EPROM mit der Bezeichnung 27CX010 hat je nach Version Zugriffszeiten von 55, 70 oder 90 ns. Ein 256-KBit-Typ, der Mitte des Jahres auf den Markt kommen soll, wird sogar nur 40 ns bieten.

Der 27CX010 stellt eine Gemeinschaftsarbeit von ICT und Advanced Micro Divices (AMD) dar. ICT hat hierbei das Chip-Design übernommen, AMD wird die Schaltkreise fertigen. Der 27CX010 soll in einer Organisation von 128 K × 8 Bit verfügbar sein und (in 1000er Stückzahlen) 98, 84 bzw. 56 Dollar kosten. *MP*

Integrierter optischer Schaltkreis

Auf der Basis eines vor vier Jahren entwickelten optischen Schalters, der Licht so zu steuern vermag wie ein Transistor den Fluß der Elektronen, wurde durch die Bell Laboratories in New Jersey ein integrierter optischer Schaltkreis entwickelt. Bis Ende der neunziger Jahre soll der optische Prozessor so weit entwickelt worden sein, daß dann Supercomputer mit der hundertfachen Leistung heutiger Rechner möglich sind. Der optische Computer ist genauso aufgebaut worden wie ein elektronischer, anstelle von Transistoren werden gläserne Linsen und Masken miteinander zu sogenannten S-SEEDs (Symmetric-Self-Electro-Optic-Effect Devices) verbunden. Sie können ihren Zustand eine Milliarde Mal pro Sekunde ändern. Die S-SEED-Elemente messen fünf tausendstel Quadratmillimeter. Die Taktfrequenz beträgt ein Megahertz.

Die Eignung optischer Bauelemente für Hochleistungsrechner ergibt sich nicht nur aus der höheren Ausbreitungsgeschwindigkeit von Lichtsignalen, sondern auch aus der leichter zu realisierenden Möglichkeit der parallelen Übertragung durch ein einziges Lichtkabel. *Fa*

Lötfreie Montage auf flexiblem Film

Die Firma Polyflex Circuits Inc., USA, entwickelte ein Verfahren, mit dem es möglich wird, Bauelemente auf einen flexiblen Polyesterfilm lötfrei zu montieren. Mit diesem als APC (Additive Polymer Circuit) bezeichneten Verfahren sollen Kosteneinsparungen von bis zu 50 % erreicht werden, indem zum einen die Zahl der Produktionsschritte, die sonst bei starren Leiterplatten erforderlich waren, drastisch verringert werden konnte und zum anderen kein Lötvorgang mehr notwendig ist.

Beim APC-Verfahren wird ein elektrisch leitendes Polymer in ein oder zwei Ebenen als Leiterbahn auf einen maximal 20 × 20 Zoll großen Polyesterfilm gedruckt. Diese Filme lassen sich auf Gehäuseflächen aufkleben. Gegenwärtig werden Linienbreiten von 0,5 mm erreicht. Es wird aber daran gearbeitet, Linienbreiten von 0,25 mm zu erreichen. Statt des Lötzinns wird ein Material aus leitendem Expoxydharz (Poly-Solder) verwendet, das von der Firma selbst hergestellt wird.

Ein Nachteil des neuen Verfahrens besteht darin, daß die Stromstärke und die Verlustleistung bei den jeweiligen Bauelementen eingeschränkt sind. *Fa*

Farb-Scanner mit Piezo-Element

Bei dem Farb-Scanner der Firma Array Technologies Inc. (USA) wurde ein Array aus Fotodioden mit einem piezoelektrischen Element verbunden. Die für die Abtastung erforderliche Bewegung des Arrays wird in präzisen Submikrometer-Schritten durch das Piezo-Element erzeugt. Mehrere aufeinanderfolgende „Mikroschritte" während eines Abtastzyklus führen zu einer Mittelwertbildung des entsprechenden Farbsignals. Mit dieser Anordnung ergibt sich für jeden Farbkanal des Scanners ein Dynamik-Bereich von maximal 12 Bit. Im Gegensatz zu anderen Scannern sind nicht mehr jeweils drei Abtastzyklen (jeweils einer für die Grundfarben Rot, Grün, Blau) erforderlich. Die RGB-Informationen werden hier jeweils in einem gemeinsamen Zyklus gewonnen. *Fa*

Unix System V im Netz

Die Prime Computer GmbH demonstierte während der CeBIT '90 auf ihrem Unix-Rechner Prime EXL als eine der ersten das neue Unix System V Release 4.0. Angestrebt wurde mit dieser Version die Integration der bisher unterschiedlichen Unix-Versionen in einem einheitlichen Betriebssystem. Bestandteil ist die grafische Benutzeroberfläche Open Look, basierend auf X-Window, deren Pop-up-Menüs und definierbare Icons höhere Benutzerfreundlichkeit versprechen.

Der EXL wurde in einer vernetzten Umgebung gezeigt. *MP*

64-Kanal-IFSS-Multiplexer

Der 64-Kanal-IFSS-Multiplexer „vervielfacht" einen seriellen Interfacekanal gemäß TGL 42886 (IFSS-Interface). Mit seiner Hilfe kann ein Personal- oder Arbeitsplatzcomputer über eine einzige serielle Schnittstelle wahlfrei mit 64 IFSS-Gegenstellen kommunizieren. Der IFSS-Multiplexer ist zur Prüffeldautomatisierung in der elektronischen Baugruppen- und Gerätefertigung vorgesehen. Er unterstützt die quasisimultane automatische Funktionsprüfung und Kalibrierung von bis zu 64 intelligenten Baugruppen oder Geräten. Der IFSS-Multiplexer ist modular als international genormtes 19 Zoll-Tischgehäuse aufgebaut. Zur Nachnutzung steht eine vollständige Konstruktionsdokumentation zur Verfügung.

VEB Elektro-Apparate-Werke „Friedrich Ebert" Berlin-Treptow, Zentrum für Forschung und Technologie, Abt. WG, Koll. Mayerhöfer, Storkower Straße 101, Berlin, 1055; Tel. 4 38 84 02 *Dr. Claßen*

Konstruktion lufttechnischer Anlagen

Für die rechnergestützte Erzeugung von Projektzeichnungen für lufttechnische Anlagen wurde das Variantenprogramm **EBROKA2D** entwickelt. Es erzielt bei vorhandenen computerlesbaren Gebäudezeichnungen enorme Zeiteinsparungen bei der Konstruktion der Lufttechnik. Folgende Merkmale charakterisieren seine Leistungsfähigkeit: TGL-gerechte automatische Konstruktion, wahlweise automatisches Aneinanderfügen der erzeugten Bauteile, Anzeige der Ergebnisparameter, Weiterarbeit an Verzweigungen in beiden Richtungen, wahlweise automatisches Erzeugen einer beliebigen zweiten Ansicht orthogonal zur ersten Ansicht.
Für die Nutzung des Variantenprogrammes sind notwendig: ein 16-Bit-Rechner mit Hauptspeicher > 512 KByte, Festplatte, Digtizer, Plotter, PCCAD oder kompatibles CAD-Programm.

VEB Sachsenring Automobilwerke, Abteilung Projektierung, PF 311-313, Zwickau, 9541; Tel. 70 32 96 *Baumann*

Module für KC 85/2 ... 4

1. AD-Modul mit Potentialtrennung
Eine Unipolare Eingangsspannung von 2 V bzw. 10 V wird in ein 12-Bit-Digitalwort gewandelt; die Potentialtrennung des Analogeinganges vom Rest der Schaltung erfolgt mittels Trennübertrager und Optokoppler. In einer speziellen Ausführungsvariante (Verhältnismessung von Spannungsteilern) kann eine Auflösung von 15 Bit bei einer Meßzeit von 200 ms erreicht werden.

2. EPROM-/CMOS-RAM-/PIO-Modul
Der Modul realisiert Programmspeicher von 16 KByte (2 × U 2764, steckbar), Datenspeicher von 2 KByte (U 6515, batteriegestützt) sowie 16(+4) Ein-/Ausgabeleitungen.
Für beide Module gilt: vollständige Anpassung an das Modulkonzept der KC 85/2 ... 4; originale Gehäuselösung der KC-Module. Die gezielte Anwendung beider Module im KC 85/2 ... 4 ermöglicht es, preiswerte Meßwerterfassungs- und/oder Steuerungsrechner aufzubauen, die keinerlei Massenspeicher benötigen bzw. durch die nichtflüchtigen Speicher und Autostart jederzeit arbeitsbereit sind. Ein weiterer interessanter Anwendungsfall ist die durch die Potentialtrennung mögliche On-line-Erfassung großer Datenmengen am Patienten in der Humanmedizin.

Fa. Dipl.-Ing. H.-J. Ollendorf, Ratiomittelbetrieb, PF 211, Brunkau, 3511 *Ollendorf*

Kontrolle biotechnologischer Prozesse

Von uns aus wurde eine Lösung zur Erfassung, Protokollierung und Auswertung von maximal 8 analogen und 6 digitalen Meßwerten einer biotechnologischen Pilotanlage zur Herstellung pektinolytischer Enzyme erarbeitet. Die Rechnerkonfiguration besteht aus einem aufgerüsteten KC 87 mit Magnetbandgerät, Drucker, Bildschirm und entsprechenden Modulen. Die Meßwerte werden zeitzyklisch erfaßt (min. 1 s), auf Ausreißer getestet und die berechneten Mittelwerte protokolliert. Das modular strukturierte Softwarepaket **PRODAT** ermöglicht eine weitgehende Änderung von Parametern und Gerätezuweisungen auch während des Betriebes. Die ausgelagerten Meßwertfiles sowie ermittelten Laborwerte können mit dem Arbeitsplatzcomputer A 7100 und dem Programmpaket **KOMDAT** ausgewertet werden.

VE Forschungsinstitut der Obst- u. Gemüseverarbeitung, Abt. AEZ, Nicolaistraße 5, Magdeburg, 3018; Tel. 22 21 12 *Dr. Lämmerzahl*

Konvertierungsprogramm UNIDAT

Das universelle Konvertierungsprogramm für grafische Daten UNIDAT kann unterschiedliche grafische Datenformate einlesen und wandelt diese in ein internes Format um. Ausgehend von diesem internen Format können folgende grafische Datenformate erzeugt werden.
Einlesen: GKS1600-Format, GKS1800-Format, CADdy-Format und ROK-Format.
Ausgeben: zusätzlich noch HPGL-Format und IGES 3.0-Format.
Durch dieses universelle Konvertierungsprogramm wird eine Datenkompatibilität zu anderen grafischen Programmsystemen ermöglicht.

VEB Robotron-Elektronik Dresden, Bereich System- und Softwareapplikation, PSF 710, Karl-Marx-Stadt, 9010; Tel. 39 73 67 *Schulz*

STEP 5-Analyzer

Mit diesem Programm wird der Nachteil behoben, daß bei Austestung komplexer PLC-Programme mit den Programmiergeräten PG 675/685 (Siemens) kein Echtzeitbetrieb gewährleistet wird. Der S-5-Analyzer wird in das auszutestende Programm eingebunden und ermöglicht auch bei Alarmbearbeitung Echtzeitbetrieb. Es können maximal 16 Kanäle belegt werden, wobei Kanal 0 als Startkanal benutzt wird. Über Voreinstellung kann festgelegt werden, ob der Analyzer extern oder über Kanal 0 mit fallender oder steigender Flanke gestartet wird.
Zur Suche sporadischer Fehler ist permanente Aufzeichnung möglich (max. 250 Signalwechsel/Zyklus). Der S 5-Analyzer ist bei allen Siemens-Automatisierungsgeräten ab AG S 5-130 W, S 5-150, S 5-115 U etc. implementierbar.

VEB Werkzeugmaschinenfabrik UNION, Karl-Marx-Stadt, Koll. Hornbogen oder Schunk, Postfach 1043, Karl-Marx-Stadt, 9010 *Dr. Aßmann/Krause*

Druckerserviceprogramme für 1157/269 am A 7150

Die im Angebot befindlichen Programme nutzen die grafischen Möglichkeiten des 1157/269 mit IFSS-Schnittstelle am A 7150 unter DCP aus. Es können mit diesen Programmen unter anderem ASCII- und Word-Star-Dateien gedruckt werden (enthaltene Drucksteuerbefehle werden ausgewertet). Zur Verfügung steht der IBM-Zeichensatz (20H-FFH). Die Zeichen ab 80H werden grafisch nachgebildet und können in einem Rasterfeld von 6 × 10 editiert werden. Ein weiteres Programm (resident) erlaubt den Druck des aktuellen Bildschirminhaltes.

VEB Wissenschaftlich-Technisches Zentrum der holzverarbeitenden Industrie Dresden, Bereich Werkstoffentwicklung und Anwendung, Zellescher Weg 24, Dresden, 8020; Tel. 4 66 23 81 *Dr. Büst*

AutoCAD-Druckertreiber für 24 Nadeln

Es wird ein Druckertreiberprogramm für die Programme AutoCAD und MultiCAD angeboten, das für den Drucker Epson LQ-1000 und kompatible 24-Nadel-Drucker eine höhere Auflösung ermöglicht. Die erreichte Auflösung beträgt beim gewöhnlichen 9-Nadel-Druckertreiber (FX-80/100) 120 Dots per Inch und kann bei 24-Nadel-Druckern mit dem ADI-Treiber auf 180 DPI gesteigert werden.
Der Treiber wurde in (Turbo-)Assembler geschrieben (27 KByte Quelle) und läßt sich auch an andere, nicht zum LQ-1000 kompatible Drucker anpassen. Der benötigte Puffer von etwa 8 KByte kann bei Vorhandensein einer EMS-Speichererweiterung dort abgelegt werden, das Programm benötigt dann nur knapp 1/2 KByte Hauptspeicher, die benutzte EMS-Seite kann wieder freigegeben werden. Unterstützt wird der Anschluß des Druckers an LPT1..4.

Akademie der Wissenschaften, Zentrum für wissenschaftlichen Gerätebau, Abt. BA, Rudower Chaussee 6, Berlin, 1199; Tel. 6 74 33 89 *Ihlenfeld*

Graph-Driver für A 7150

Es wird ein Driver angeboten, der es gestattet, den vollen Umfang der Grafik-Prozeduren der Unit Graph von Turbo-Pascal 5.x im hochauflösenden Grafikmodus (mit 640 × 480 Pixeln und 8 Graustufen) zu nutzen. Vorhandene Quellprogramme können damit nahezu ohne Änderungen auf dem A 7150 implementiert werden. Der Driver wird unter der Driver-Nummer 10 (= PC3270) mit *InitGraph* aktiviert. Ein analoger Driver für den EC 1834 ist in Vorbereitung. Außerdem werden Zusatz-Units angeboten, die allgemein nützliche Grafik-Funktionen realisieren, wie: Weltkoordinaten, Fensterselektionen, grafische Pop-up-Prozeduren, Fenster-Speicherungen, Hardcopy-Prozeduren, Steuerung des Grafik-Kursors mit Tastatur und Tablett.
Weiterhin ist ein Driver zur Ansteuerung eines HP-kompatiblen Plotters (SPL 430 o. ä.) unter Turbo-Graph vorhanden.

Zentralinstitut für Physik der Erde, Bereich I, Telegrafenberg, Potsdam, 1560; Tel. 31 02 34 *Köhler*

F-Tastenbelegung

In MP 1/90, Seite 26, wurde eine Möglichkeit der Tastencodierung für 16-Bit-PCs unter Nutzung der ANSI-Steuersequenzen vorgestellt. Eine nachnutzungsfähige Softwarelösung für Anwender mit geringeren Erfahrungen auf diesem Gebiet ist das Programm **TAST**. Es wurde unter Turbo-Pascal 5.0 entwickelt und bietet folgende Möglichkeiten: dialoggesteuerte Eingabe und Korrektur von Befehlsketten von bis zu 30 Zeichen Länge für je eine der Funktionstasten F1 – F10, wahlweises Stornieren oder Aktivieren von RETURN für jede Taste, Speichern und Reaktivieren jeder gewünschten Codierung, Befehlsverkettung bei Nutzung des stornierten RETURNS, globale und kontextbezogene Hilfestellungen zu Befehlseingabe, -sicherung und -aktivierung sowie einigen Tips und Tricks für die Anwendung, die im Programm jederzeit abrufbar sind. Zur Übernahme ist eine mit 360 KByte formatierte Diskette notwendig.

Jürgen Plesse, Aufbauweg 2, Köllitsch, 7291

Hinweis

Der Verlag Technik bittet um die folgende Berichtigung in dem Buch Claßen/Wiesner: Wissensspeicher 80286-Programmierung.
In der Befehlsliste des Mikroprozessors 80286 (Anhang A) sind Seiten vertauscht worden.

Befehle	Richtige Fortsetzung
Datentransferbefehle	S. 2..
Arithmetikbefehle	S. 21.
Logikbefehle	S. 215
Programmsteuerbefehle	S. 213, 214, 211

Datenanalyse mit dem Personalcomputer

von W. Fleischer und M. Nagel, Verlag Technik, Berlin 1989, 368 S., 46 Bilder, 87 Tafeln

Es gibt wohl kaum ein Gebiet der experimentellen Natur-, Technik- und Sozialwissenschaften, in dem nicht aus anfallenden Datenmengen Aussagen, Wertungen oder Entscheidungen getroffen werden müssen. Dafür gibt es umfangreiche Fachliteratur, die die Methoden und Verfahren beschreibt und durch die entwickelte Computertechnik vielfältige Möglichkeiten ihrer praktischen Umsetzung und effektiven Anwendung. Demgegenüber ist in der Praxis oft festzustellen, daß diese Möglichkeiten mangels Sachkenntnis, Überblick und Einarbeitungszeit viel zu wenig genutzt werden. Diesem Mangel abzuhelfen und die Möglichkeiten der statistischen Datenanalyse anwendungsbereit zu propagieren, ist Anliegen des vorliegenden Buches. In leicht zugänglicher Darstellung enthält es eine Vielzahl von Verfahren und Methoden zur Auswertung vorwiegend ein- und zweidimensionaler Datensätze mit statistischen Methoden, wobei insbesondere der grafischen Darstellung eine deutliche Aufmerksamkeit zuteil wird. Nach zwei einführenden Kapiteln, in denen die wesentlichen Grundbegriffe und Methoden der praktischen statistischen Datenanalyse mit Hilfe zahlreicher instruktiver Beispiele anschaulich erläutert werden, erfolgt in zwei weiteren Kapiteln die Darlegung der Grundlagen der Wahrscheinlichkeitstheorie, der mathematischen Statistik und Testtheorie und ihrer Anwendungsmöglichkeiten. Schwerpunkte im weiteren sind die Varianzanalyse und die lineare Regression, die zunächst für zwei Zufallsvariablen eingeführt und erläutert, und im letzten Teil des Buches zur multiplen Regression geführt wird. Ein besonderes Kapitel ist der Transformation von Datensätzen zur Erhöhung der Effizienz der statistischen Analyse gewidmet. Jeder thematische Abschnitt behandelt zunächst die Grundlagen, demonstriert sie an einem oder mehreren Beispielen und zeigt schließlich an einem entsprechenden Computerprogramm die rechentechnische Umsetzung der dargelegten Verfahren. Das führt einerseits den Lernenden auf einfachem Wege von der Theorie zur Praxis und erzeugt andererseits eine gewisse Autonomie der einzelnen Kapitel, was das Buch besonders zum selektiven Studium und zum Nachschlagen geeignet erscheinen läßt. Ein Anhang, der knapp ein Viertel des Gesamtumfanges einnimmt, enthält die Quelltexte der Basic-Programme, die im Sachteil des Buches erläutert werden. Ist dies auch – gemessen an der Bereitstellung einer entsprechenden Programmdiskette – eine Notlösung, so wird derjenige, der sich der Mühe unterzieht, das eine oder andere Analyseprogramm einzutippen, neben einem Wissenszuwachs auch die direkte Möglichkeit praktischer Datenanalyse aus der Arbeit mit dem vorliegenden Titel gewinnen.

Das Buch ist expressis verbis für den Praktiker geschrieben und allen jenen zu empfehlen, die sich rasch in die statistische Analysentechnik mit dem Ziel ihrer unmittelbaren praktischen Anwendung einarbeiten wollen oder müssen. Darüber hinaus kann es auch durch seine fachliche Breite dem bereits auf diesem Gebiet Tätigen nützliche Hinweise geben und zum Nachschlagen und zur Vertiefung seiner Kenntnisse und Fähigkeiten dienen. *Prof. Dr. W. Schwarz*

PC-, MS-DOS-Schulung

Programmierte Unterweisung in den Betriebssystemen PC-DOS und MS-DOS

von G. Renner, Markt-und-Technik-Verlag, Haar bei München, 1990, 490 S., 98,– DM, incl. 1 Diskette, ISBN 3-89090-777-6

Noch ein Buch zu MS-DOS. Bringt es wirklich etwas Neues? Inhaltlich wird man kaum Neuigkeiten über die DOS-Benutzerschnittstelle aus einem Buch erfahren. Die Frage ist also, wie das Wissen „rübergebracht" wird. Durch die äußere Aufmachung wird der Einsatzzweck schon angedeutet. Es handelt sich nämlich weniger um ein Buch, als vielmehr um einen robusten Ordner mit Einzelblättern, in denen gewöhnlich die Systemliteratur mit der Hardware ausgeliefert wird. Zum intensiven Gebrauch bestimmt. Das Werk gliedert sich in einen Einführungsteil (380 Seiten) und eine Referenz (100 Seiten). Diese stellt auch verwöhnte Ansprüche zufrieden. Sie ist übersichtlich und – soweit das überhaupt einschätzbar ist – vollständig. Alle Befehle der Versionen 3.3 und 4.0 sind aufgenommen.

Der Lernteil, der sich als „programmierte Unterweisung" versteht, ist in Lektionen gegliedert, an deren Ende mit Fragen und zugehörigen Antworten eine Wiederholung der wichtigsten Fakten vorgenommen wird. Besonderer Wert wird auf die Erklärung aller Begriffe gelegt, die erfahrungsgemäß dem Einsteiger einige Schwierigkeiten bereiten, wobei auch anspruchsvolle Themen (EMS, Cache) zum Ende des Buches nicht ausgespart werden.

Man kann dem Buch eine sehr umfassende Darstellung attestieren. Einführend werden die PC-Hardware, allgemeine Aufgaben des Betriebssystems und die Entwicklung von MS-DOS über die einzelnen Versionsetappen behandelt. Im Mittelpunkt stehen dann die residenten und transienten Systembefehle. Der Drang nach Vollständigkeit geht gar so weit, daß neben DEBUG auch der völlig obsolete Zeileneditor EDLIN aufgenommen wurde. Statt des starren Festhaltens am Systemumfang von Microsoft wäre vielleicht die Aufnahme einiger allgemein benutzter Dientsprogramme und Utilities hilfreicher gewesen.

Um dem unbedarften Nutzer die Möglichkeit zu geben, sich ohne fremde Hilfe die notwendigen Fertigkeiten anzueignen, sind wichtige Bedienalgorithmen (Systemstart, Konfigurieren, Formatieren) Schritt für Schritt als Kochrezept vorgegeben. An manchen Stellen erscheint mir der Formalismus dann aber doch ein wenig übertrieben (Frage: Welche Teilaufgaben unterscheiden wir bei einer Arbeitssitzung? Anwort: Wir unterscheiden die Aufgabe vorbereiten, die Aufgabe bearbeiten und die Aufgabe nachbereiten).

Obwohl als Lehrbuch konzipiert, kann es ebenso zum Nachschlagen verwendet werden. Dafür stehen die zahlreichen relevanten Beispiele, eine übersichtliche Gliederung, Inhalts- und Indexverzeichnis. So lassen sich auch Handgriffe meistern, die nicht jeden Tag benötigt werden.

Natürlich dürfen Batch-Dateien nicht fehlen. Ihnen ist eine ganze Lektion gewidmet. Neben der Beschreibung aller Steuerbefehle gibt es zwei Dutzend ausführliche Beispiele. Und damit dem noch unsicheren Einsteiger das Abtippen erspart bleibt, sind – wie es guter Brauch ist – alle Programme auf der zugehörigen Diskette. Vielleicht weiß der Autor auch, daß mit EDLIN so schnell kein Erfolg zu erzielen ist.

Fazit: Haben Sie zu Ihrem Rechner einen Satz guter Systemliteratur erhalten, so läßt sich auf das vorliegende Buch sicher verzichten. Sieht es auf diesem Gebiet aber eher spärlich oder englischsprachig aus, oder schlagen Sie sich gar mit einer südostasiatischen Übersetzung herum, so sei gerade dieses Werk als Ergänzung empfohlen. *U. Schulze*

Aufbau und Arbeitsweise von Rechenanlagen

von W. Coy, Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig 1988, 275 Seiten

Das vorliegende Buch stellt eine ausgezeichnete Einführung in die Organisation und Architektur moderner Rechenanlagen dar. Im ersten Teil des Buches sind fünf Abschnitte zum Schaltungsentwurf und zur Rechnerorganisation enthalten. Hier findet der Leser alle neuen Erkenntnisse von der Schaltungslogik über integrierte Schaltkreise bis hin zum Entwurf von sequentiellen Maschinen zusammengefaßt und verständlich erklärt.

Im zweiten Teil des Buches, dem Hauptteil, wird der Rechner auf Architekturebene behandelt. Es werden Datendarstellungen, Aufbau und Arbeitsweise einer Zentraleinheit, Befehlssätze und Mikroprogrammierung, Assemblerprogrammierung, arithmetische Operationen, Bussysteme, E/A-Organisation und moderne, hierarchisch aufgebaute Speicherorganisationen, einschließlich virtueller Speichertechnik, Cachespeicher und Paging, beschrieben.

Der dritte Teil ergänzt das Buch mit grundlegenden Aussagen zu Betriebssystemen, die gewissermaßen eine Erweiterung der Hardware darstellen. Neben einer Einführung in die Aufgaben eines Betriebssystems werden vor allem Prozeßverwaltung, Speicherverwaltung und Kommandosprachinterpreter behandelt. Sehr wertvoll ist auch der abschließende Abschnitt, der einen hervorragenden Ausblick auf die Entwicklungstendenzen in der nahen Zukunft gibt.

Im zweiten Teil des Buches wird bei der Architekturbetrachtung die Verbindung zu Schaltkreissystemen wie MC 68000, Intel 8086/80286 oder typischen Kleinrechnerarchitekturen, z. B. VAX 11/780, hergestellt, während für den dritten Teil als Basis zur Betriebssystembetrachtung das Betriebssystem UNIX gewählt wurde.

Auf Grund des Lehrbuchcharakters mit Übungsaufgaben zur Selbstkontrolle ist das Buch als Zusatzliteratur zum Informatik-Grundstudium geeignet.

Natürlich sind bei der Breite des Buches kleinere Unexaktheiten nicht ganz ausgeschlossen. So wird der Massbus als synchrone Variante des VAX-Ein-/Ausgabebusses eingeordnet, obwohl er lange vor der VAX-11/780 als Bus mit asynchroner Steuerung zum Anschluß großer Plattenspeicher an die PDP-11/70 entwickelt wurde und heute seine Bedeutung für die VAX praktisch verloren hat. Die größeren Ungenauigkeiten sind im Betriebssystemteil zu finden. So entspricht u. a. das Schichtenmodell nicht dem heute üblichen Stand. Auch werden die POSIX-Standardentwürfe fälschlicherweise als UNIX-Standards charakterisiert. Die POSIX-Standards (Portable Operating System Interface) des IEEE verfolgen die Zielstellung der Erarbeitung von Betriebssystemstandards zur Implementierung von portablen Betriebssystemschnittstellen auf Nicht-UNIX-Systemen unter gleichzeitiger Wahrung der Kompatibilität zu UNIX-System V und X/OPEN. *Prof. Dr. Horn*

Informationsschrift zu LANs mit EC 1834

Die Informationsschrift beinhaltet einführend den internationalen Stand auf dem Hardwaresektor und der Netzbetriebssysteme und die bisher in der DDR durchgeführten Arbeiten. Schwerpunkt ist die Behandlung von praktischen Fragen des Aufbaus und der Funktionsweise von LAN-Controllern und ihres Einsatzes im EC 1834, des Zusammenwirkens von LAN-Controller und Netzwerkbetriebssystem mit der Kommunikationsschnittstelle NETBIOS und der Inbetriebnahme und des Tests eines LANs.

Im Anhang ist eine Zusammenstellung von Begriffserklärungen für das Arbeitsgebiet enthalten. Damit vermittelt die Informationsschrift den Projektanten und Nutzern von lokalen Netzen notwendiges Basiswissen und unterstützt vor allem Spezialisten beim konkreten Aufbau und der Inbetriebnahme des LAN.

Der Preis für die Informationsschrift (ca. 80 Seiten) beträgt 250,00 M.

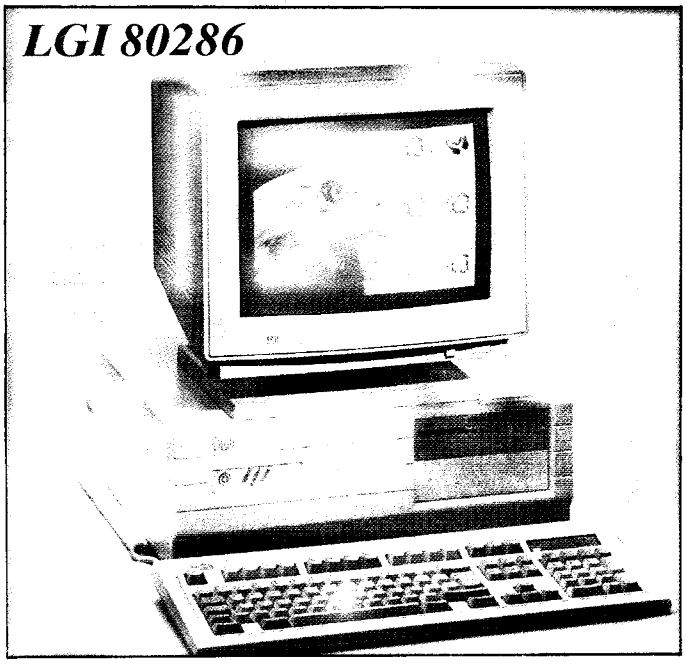
Bestellungen richten Sie bitte an: DVZ Suhl, Bereich Absatz und Beschaffung, Am Fröhlichen Mann, Suhl, 6017

Leserumfrage

Liebe Leserinnen und Leser!

Um noch besser beurteilen zu können, wo die gegenwärtigen und künftigen Schwerpunkte Ihres Interesses liegen, veranstalten wir auf dieser Seite eine kleine Umfrage.
Wir bitten Sie, mit dem Ausfüllen des Fragebogens das künftige Profil der MP so mitzubestimmen, daß es Ihren Vorstellungen näherkommt. Schon das wäre sicher die kleine Mühe des Ausfüllens und Absendens wert. Zusätzlich erhalten Sie damit die Gelegenheit, an einer Verlosung teilzunehmen, für die uns einige Firmen freundlicherweise Preise zur Verfügung stellten. Als Hauptpreis winkt ein Personalcomputer von LGI, den wir Ihnen auf dieser Seite vorstellen. Weitere Preise sind von der Firma Volkswriter Software Deutschland GmbH, Eschenried: 20 Textverarbeitungsprogramme „Volkswriter" und zahlreiche Lernprogramme, beispielsweise für MS-DOS und zum effektiven Umgang mit Ihrem PC. Außerdem verlosen wir Computerliteratur.
Einsendeschluß für die Teilnahme an der Verlosung, die unter Ausschluß des Rechtsweges erfolgt, ist der 30. August 1990. Die Gewinner werden dann persönlich benachrichtigt.
Also: Fragebogen ausfüllen und an folgende Adresse senden:
Verlag Technik
Redaktion MP
Kennwort Leserumfrage
Oranienburger Straße 13/14
Berlin, 1020

Zu gewinnen:



Die Produktpalette des südkoreanischen Computerherstellers Lucky GoldStar International ist recht umfangreich und geht dabei von 80286ern über 80386er bis hin zu PCs der 80486er Klasse. Ebenfalls im Angebot sind der Laptop LGI 286 LT sowie SuperScan- und VGA-Monitore, aber auch Optical und Floptical Disks.
Im folgenden stellen wir Ihnen die Reihe der 80286er PCs etwas näher vor, zu der auch unser Hauptpreis gehört.
Die zwei auf der Basis des Intel-Prozessors 80286 angebotenen PCs unterscheiden sich nur durch die Taktfrequenz. Der LGI 286n wird mit maximal 12 MHz und der LGI 286s mit maximal 16 MHz angeboten. Beide PCs können auch mit 8MHz betrieben werden. An der Vorderseite des Grundgerätes sind neben den Kontrolleuchten auch Funktionsschalter bzw. Tasten. Mit dem Schlüsselschalter (links) können Sie sämtliche Daten auf der Festplatte vor fremdem Zugriff schützen, da hiermit die Tastatur blockiert werden kann.
Die Hauptplatine des LGI 286x ist ein New Enhanced AT-Motherboard (NEAT), dessen entscheidender Unterschied zu anderen Technologien die softwaremäßige Einstellung verschiedener Systemparameter ist. Kein Aufschrauben und kein Stellen von Mikroschaltern mehr. Die wichtigsten technischen Daten sind ein RAM von mindestens 1 MByte, ein 5 1/4-Zoll-Laufwerk (1,2 MByte/360 KByte), eine Festplatte (20 oder 40 MByte), eine hochauflösende Farbgrafik (bis 1024 × 768 Punkte, VGA) sowie serielle und parallele Schnittstellen. Eine Erweiterung um ein zweites Diskettenlaufwerk (5 1/4- oder 3 1/2-Zoll) und eine Ergänzung mit dem mathematischen Koprozessor 80287 sind möglich. Als Software wird standardmäßig MS-DOS 3.3 einschließlich eines speziellen LGI-Lernprogramms mitgeliefert, ebenso Microsoft GW-Basic und das LGI-Software-System.

Technische Daten

Prozessor	Intel 80286, 8/12 MHz bzw. 8/16 MHz, Sockel für optionalen Koprozessor Intel 80287
Hauptspeicher	Standardmäßig 1 MByte RAM auf der Hauptplatine, auf 8 MByte RAM erweiterbar
Massenspeicher	Floppylaufwerk 1,2 MByte, 5,25 Zoll Festplatte 20 MByte oder 40 MByte
Video Controller	VGA-Controller 16 Bit mit Analog- und TTL-Signal, maximale Auflösung: 1024 × 768 Punkte
Tastatur	102 Tasten, erweiterte AT-Tastatur (IBM-kompatibel)
Schnittstellen	zwei serielle (RS232C) und eine parallele
Erweiterungsslots	5 × 16 Bit und 3 × 8 Bit
Maße (B × T × H)	433 × 420 × 155 mm³

- Ich bin

	beruflich	privat
Computeranwender	O	O
Hardwareentwickler	O	O
Softwareentwickler	O	O
wissenschaftlich tätig	O	
Schüler/Lehrling/Student	O	

Sonstiges bitte hier eintragen (z. B. Führungskraft, DV-Mitarbeiter, Interessent):

. . . .

- Ich bekomme die MP
  - im Abonnement O
  - im Freiverkauf O
  - dienstlich O

- Sollten Sie die MP dienstlich im Umlauf erhalten, geben Sie bitte, soweit bekannt, noch die Anzahl der weiteren Leser entsprechend den folgenden Kategorien an:
  - Computeranwender ....
  - Hardwareentwickler ....
  - Softwareentwickler ....
  - wissenschaftlich Tätige ....
  - sonstige Mitarbeiter ....

- Die Beiträge in der MP sind überwiegend
  - zu populärwissenschaftlich O
  - zu wissenschaftlich-theoretisch O
  - zu fachspezifisch O
  - verständlich und ausgewogen O

- **Welche Themen gehören nicht in die MP bzw. sollten reduziert werden?**

. . . .

- Welche Themen fehlen in der MP gänzlich oder sind zu gering vertreten?

. . . .

- Weitere Hinweise, Kritiken, Wünsche:

. . . .

Absender

Name . . . .

Vorname . . . .

Straße, Nr. . . . .

PLZ, Ort . . . .

Leipziger Frühjahrsmesse 1990

War die Leipziger Frühjahrsmesse im 825. Jahr des Bestehens der Leipziger Messen nun die letzte? Sicher nicht. Aber Ausstellern und Besuchern der Computerbranche drängte sich angesichts der Veränderungen in der DDR und des „Zusammenwachsens" doch die Frage nach der Berechtigung auf; und zwar im Hinblick auf die bereits kurz darauf veranstalteten Messen CeBIT 90 als weltgrößte Computermesse und Industrie '90, jeweils in Hannover. Anzeichen für mangelndes Interesse an der LFM ließen sich allerdings noch nicht erkennen. Das 340 000 Quadratmeter große Ausstellungsterrain war restlos vermietet worden; davon mehr als ein Drittel an ausländische Unternehmen aus etwa 70 Ländern. Gegenüber der Vorjahres-LFM waren das etwa 8000 Quadratmeter zusätzlich.

Hinsichtlich des Niveaus der Produkte dokumentierte die LFM den internationalen Stand wiederum nur teilweise. Westlichen Firmen waren bekanntlich noch Beschränkungen zur Lieferung von Hochtechnologien auferlegt; RGW-Länder nutzen traditionell ihre Heimmessen zur Präsentation von Neuheiten, die offenbar rar sind. So zeigten sie in Leipzig zum ...zigsten Mal ihren PC Elwro 801 AT (Polen), PP 06 (CSFR) oder Felix PC (Rumänien).

Robotron nutzte seine Heimmesse, um nicht nur den bereits im Vorjahr erwarteten EC 1835 vorzustellen, sondern gleichzeitig zwei Varianten von ihm: den 32-Bit-EC 1835 und einen Transputer-EC 1835. Außerdem die neue EC-1834-Version EC 1834.01. Unverkennbar eine Bescheinigung der Erneuerungs-, Entschuldigung: Innovationsrate bei Robotron. Im folgenden haben wir für Sie wieder die Neuheiten zusammengestellt, die uns erwähnenswert erschienen.

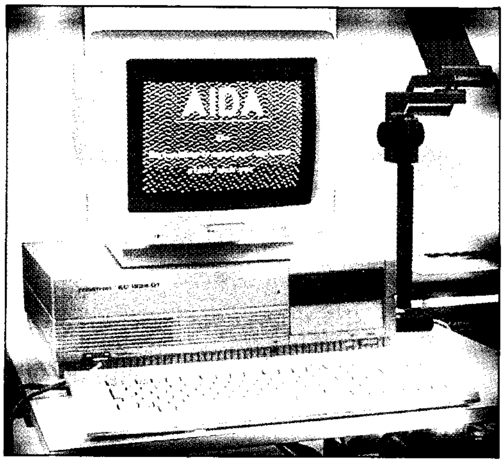


Bild 1

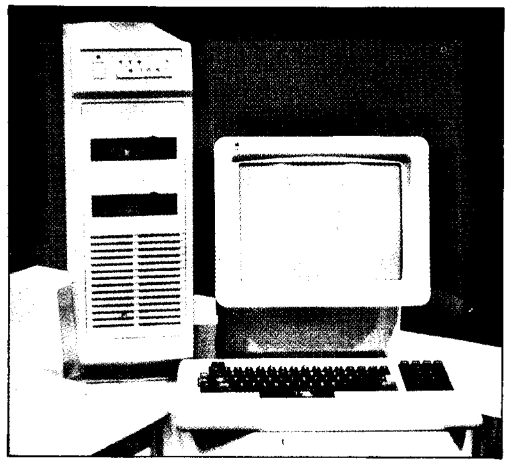


Bild 2

Hardware

- **PCs und Workstations**

Da wäre zunächst der bereits genannte **EC 1834.01** (Bild 1). Obwohl Robotron versucht hat, stärker auf Bauelemente aus der DDR und dem RGW zu orientieren, wurde im Unterschied zum zwei Jahre älteren Vorgänger, EC 1834, nach Angaben von Robotron die „direkte" XT-Kompatibilität erreicht. Auch der EC 1834.01 arbeitet mit dem 8086-kompatiblen Prozessor K 1810 WM86 aus der UdSSR. Seine Taktfrequenz ist 4,9 MHz. Für den nun standardmäßig auf der Systemplatine vorhandenen 640 KByte großen RAM werden 64-KBit- und 256-KBit-Schaltkreise verwendet. Für Erweiterungen stehen 8 Steckplätze zur Verfügung (4 bis 6 indirekte und 2 bis 4 direkte Steckverbinder), von denen jedoch mindestens 4 für Harddisk- und Floppy-Adapterkarte, für Videoadapter und für eine serielle oder parallele Schnittstellenkarte belegt werden. Die noch freien 4 Steckplätze können mit der vom 1834 bekannten 384-KByte-RAM-Karte, mit der Beschleunigerkarte BK 600 (vgl. MP 2/90, 3. US), mit einer LAN-Adapterkarte oder mit anderen XT-kompatiblen Karten belegt werden. Der Floppydisk-Adapter kann vier 5,25-Zoll-Laufwerke mit einer Kapazität von je 720 KByte ansteuern; ausgestattet ist der 1834.01 mit einem Laufwerk. Der Harddisk-Adapter kann zwei Laufwerke mit maximal je 63 MByte ansteuern. Er verwendet dafür den Controller U 82062, der eine Datenübertragungsrate von 5 MBit pro Sekunde gewährleistet. Er erreicht theoretisch einen Interleavefaktor von 3, mit dem Betriebssystem DCP von Robotron jedoch nur von 6. Als Monitoradapter wird von Robotron zwar standardmäßig nur ein alphanumerischer Monochromadapter angeboten, daneben sind aber auch noch ein monochromer Grafikadapter (MGA), ein CGA-kompatibler Adapter und ein VGA-Adapter verfügbar (im Gegensatz zum EC 1834 zu Lasten von nur einem Steckplatz). Als Betriebssysteme stehen DCP 3.3, MS-DOS 4.01 und das Unix-kompatible Mutos 1834 zur Verfügung.

Weiterhin bot Robotron in Leipzig den Nachfolger des XTs, den AT-kompatiblen **EC 1835** an. Diesen Rechner haben wir bereits im vorigen Heft näher vorgestellt. Für anspruchsvollere Aufgaben, wie Bildverarbeitung, CAD oder komplizierte Berechnungen, waren noch zwei „Verwandte" des EC 1835 präsent: die Transputer-Workstation **EC 1835-TR8** und der 386er AT **EC 1835 turbo**, beide Geräte finden Sie auf der 4. Umschlagseite.

Viele Besucher erwarteten, daß Robotron zur LFM einen Nachfolger des monumentalen K 1840, den K 1820 präsentiert, den Projektleiter Dr. Walter den Anwendern „als kleinen Würfel an den Arbeitsplatz bringen" will (s. MP 10/89, S. 290). Gezeigt wurde jedoch lediglich mit dem Volumen des K 1840 die Weiterentwicklung K 1845, die vor allem einen größeren Hauptspeicher besitzt. Rechner dieser Kategorie – also Minirechner oder Rechner der mittleren Datentechnik – waren zur LFM nur schwach vertreten. IBM Deutschland zeigte hier zum Beispiel ihr System /400, über das wir bereits mehrfach in MP berichteten. Bei Siemens waren mehrere Anlagen dieser Kategorie zu sehen, wobei sich die Grenzen zwischen Minis, Workstations und Hochleistungs-Personalcomputern heute nicht mehr so exakt bestimmen lassen. Das Siemens-Mehrplatzcomputersystem **MX 300** läuft unter SINIX und soll demnächst voll Unix-kompatibel sein. Als Prozessor wird hier – etwas unüblich – der 32-Bit-Prozessor NS 32332 (von National Semiconductor) mit 15 MHz Taktfrequenz verwendet. Weitere Daten: Multibus I, 4 bis 16 MByte Hauptspeicher, 2 × 200 oder 2 × 300 MByte Festplattenkapazität, 5,25-Zoll-Floppy, Streamer und Grafikmonitor.

Ebenfalls zu den Workstations zu rechnen ist der 16-Bit-Rechner **CM 1425** der Kiewer Produktionsvereinigung Elektronmasch (Bild 2). Der CM 1425 besitzt eine diskret aufgebaute CPU und eine Unibus-Schnittstelle, was Kompatibilität mit der DEC-Technik vermuten läßt. Seine Speicherkapazität kann zwischen 0,5 und 4 MByte betragen; er soll eine Leistung von 3 MIPS aufweisen. Als Massenspeicher sind eine 5,25-Zoll-Festplatte mit 11 oder 31 MByte und eine 5,25-Zoll-Floppy mit 0,5 MByte vorhanden. Der Grafikbildschirm arbeitet mit einer Auflösung von 520 × 512 Bildpunkten. Als Betriebssystem wird das Echtzeitsystem OS RW M – ein Nachfolger von OS RW – genutzt.

Videoton hatte das neueste Modell seiner VT 32-Reihe mitgebracht. Über die Rechner VT 310 und VT 320 (mit Motorola 68010-kompatiblem 16/32-Bit-Prozessor) hatten wir in MP schon berichtet; diesmal war das Modell **VT 3200** zu sehen (Bild 3), das mit Prozessoren kompatibel zum 68020/68030 arbeitet.

Die Bull AG zeigte unter anderem den Supermikrocomputer **DPX 2000** (Bild 4), ein Multiprozessorsystem, bei dem alle Kommunikationscontroller über eigene Steuerprozessoren und Datentransferspeicher verfügen.

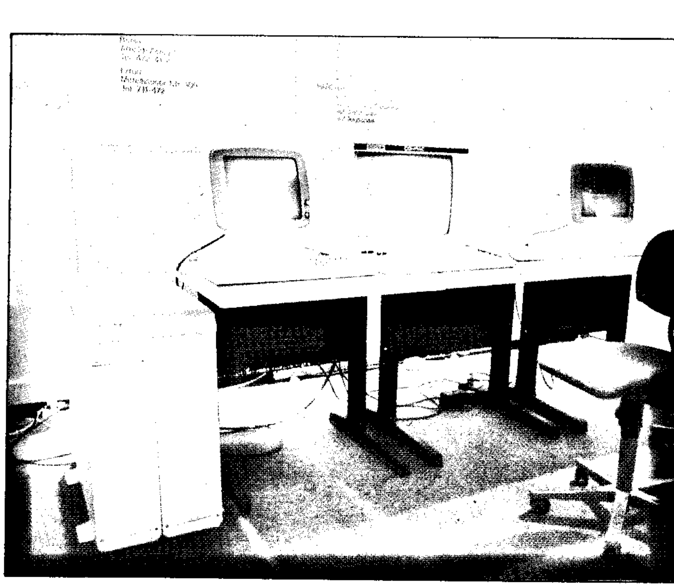


Bild 3

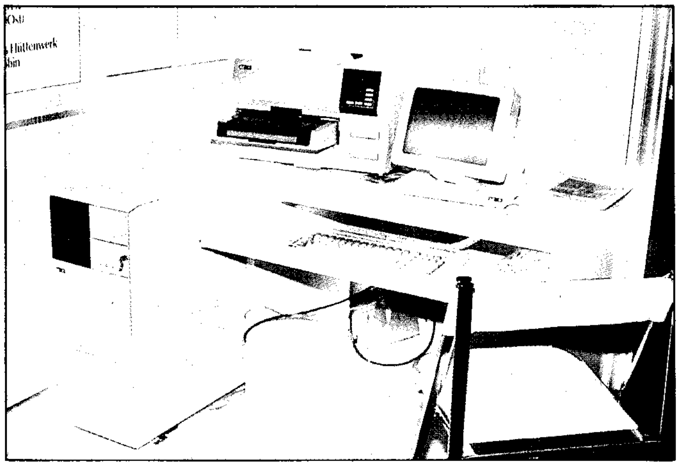


Bild 4

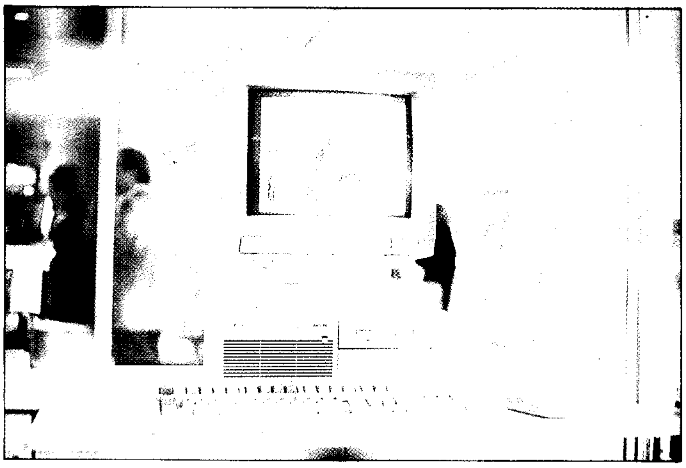


Bild 6

Bild 5

Dazu kommt ein Doppelbussystem, bei dem dedizierte 32-Bit-Hochleistungsbusse für die Kommunikation zwischen CPU – ein 32-Bit-Motorola-68020 – und Hauptspeicher einerseits und für den Datentransfer zwischen CPU und E/A-System andererseits eingesetzt werden. Jedes Basissystem läßt sich bis auf 32 oder 64 Benutzerterminals ausbauen. Die Leistung des Systems soll bis auf 5 MIPS gebracht werden können. Als Betriebssystem wird das auf Unix V basierende SPIX angeboten.
Die Präsentation von Hochleistungs-PCs hielt sich zur diesjährigen LFM in Grenzen; vieles war bereits von 1989 oder gar 1988 bekannt. Jedoch gab es eine Ausnahme: Commodore zeigte neben seinen kompatiblen Modellen PC 50-II und PC 40-III sowie neben dem bekannten Amiga 2000 erstmals öffentlich – also noch vor der CeBIT – sein neues Flaggschiff Amiga **2500/30** (Bild 5), als Basis für professionelle Anwendungen wie zum Beispiel DTP, Desktop Presentation und Desktop Video. Diese Videoleistungen wurden den Besuchern eindrucksvoll demonstriert. Herz des neuen Rechners ist der mit 24 MHz getaktete Motorola 68030, der bisherige Memory-Management-Chip ist hier bereits integriert (als Positivum sei vermerkt, daß trotz des Messetrubels das Rechnergehäuse bereitwillig entfernt wurde, so daß wir einen Blick ins Innere und auf den Prozessor werfen konnten – was wir uns bei Robotron bezüglich des EC 1835 Turbo auch wünschten, leider ohne Erfolg). Mit diesem schnellen Prozessor und durch den auf 32 Bit verdoppelten Datenbus soll der neue Amiga doppelt so schnell wie das bisherige Spitzenmodell sein. Weitere technische Angaben können Sie unserem Heft 3/1990, S. 95, entnehmen. Die Preisempfehlung für den Amiga 2500/30 liegt bei etwa 9000 DM. (Inzwischen hat Commodore bereits sein neues Modell Amiga 3000 präsentiert.)
Schon mehrfach hatten wir auf PCs mit dem Intel-Prozessor 80386 SX hingewiesen, die den schnellen 286er ATs Konkurrenz machen und als Einstieg in die 32-Bit-Welt dienen sollen. Commodore hatte ein solches Modell zur Systems 89 vorgestellt (s. MP 2/90, S. 63) und nun zur LFM mitgebracht. Der **PC 50-II** soll in seiner Einstiegsversion mit 40-MByte-Festplatte für gut 7200 DM angeboten werden; eine Version mit 100-MByte-Harddisk ist jedoch auch im Angebot. Der Rechner hat standardmäßig ein 3,5-Zoll-Diskettenlaufwerk und einen Einbauplatz für ein 5,25-Zoll-Diskettenlaufwerk. Im Gerät gibt es für Erweiterungskarten fünf Steckplätze.
Einen SX-Rechner hatte aus ihrem breiten PC-Spektrum ebenfalls die Philips AG ausgestellt. Auch der **P 3345** (Bild 6) ist mit 40- oder 100-MByte-Festplatte lieferbar und besitzt standardmäßig ein 3,5-Zoll-Diskettenlaufwerk. Weitere Daten: 1 MByte Hauptspeicher, je eine serielle und parallele Schnittstelle, VGA-Controller, 4 freie Steckplätze.
Atari präsentierte zur LFM einen großen Teil des Produktspektrums. So gab es den zur vorjährigen CeBIT vorgestellten Portfolio (s. MP 12/89, S. 384) ebenso zu sehen wie ein neues AT-Einstiegsmodell, den ABC 286-30 (über den wir in einem der nächsten Hefte berichten werden). Nach Leipzig kam auch der mobile ST, Ataris Laptop **Stacy** (Bild 7). Präsentiert wurde er erstmals zur CeBIT 89 mit 1 MByte Hauptspeicher, nun gab es die neue Variante mit 4 MByte RAM, den Stacy 4. Das Besondere an diesem Laptop ist, daß die Eigenschaften, die Musiker an den Atari STs schätzen, auf den mobilen ST übertragen wurden. Das heißt, der Stacy ist vollkommen ST-kompatibel, so daß zum Beispiel mehr als 200 Midi-Programme genutzt werden können. Durch das MIDI-Interface kann das Gerät auch vor Ort, das heißt im wechselnden Studio oder auf der Bühne problemlos genutzt werden. Zwar ist der Stacy netzunabhängig, da jedoch keine Akkus, sondern Batterien verwendet werden (jeweils 12 Babyzellen), ist eine Steckdose in Arbeitsplatznähe doch empfehlenswert. Neben dem 4-MByte-RAM besitzt der Stacy 196 KByte ROM, ein 3,5-Zoll-Floppylaufwerk (720 KByte) und eine 40-MByte-Festplatte. Der Bildschirm ist ein hintergrundbeleuchtetes Supertwist-LCD mit 600 × 400 Bildpunkten. Zur Arbeit unter der GEM-Bedienoberfläche wurde in die Tastatur bereits die Maus „umgedreht" eingebaut, als Trackball also. Es gibt jedoch auch eine Schnittstelle für eine herkömmliche Maus (der Trackball wird dann abgeschaltet) ebenso wie für zusätzliche Monitore (monochrom oder color), Drucker oder weitere Festplatten bzw. Floppystationen. Das Gewicht des Stacy liegt bei knappen 8 kg; der Preis bei etwa 6500 DM.

- **Heimcomputer**

Hier soll vor allem der Neue aus Mühlhausen genannt werden. Der **KC compact** (Bild 8) des VEB Mikroelektronik Mühlhausen ähnelt dem Bildungscomputer; das heißt, er besteht aus dem Grundgerät mit integrierter Tastatur und einem separaten Beistellnetzteil. Die Bildausgabe erfolgt über HF oder RGB auf Schwarzweiß- oder Farbfernsehgeräten bzw. auf Monitoren in drei Grafikauflösungsvarianten: 200 × 160 Bildpunkte bei 16 aus 27 Farben, 200 × 320 Bildpunkte bei 4 aus 27 Farben oder 200 × 640 Bildpunkte bei 2 aus 27 Farben. Auch kann zwischen 40 und 80 Zeichen/Zeile gewählt werden; die Zeilenanzahl beträgt stets 25. Als Prozessor wird auch beim compact der 8-Bit-Typ U 880 verwendet. Die Größe des Hauptspeichers beträgt 64 KByte, wovon etwa 42 KByte für den Anwender nutzbar sind. Im 32 KByte großen ROM sind das Betriebssystem und Basic enthalten. Zur Tonerzeugung ist ein Soundgenerator mit 3 Kanälen in 8 Oktaven vorhanden.

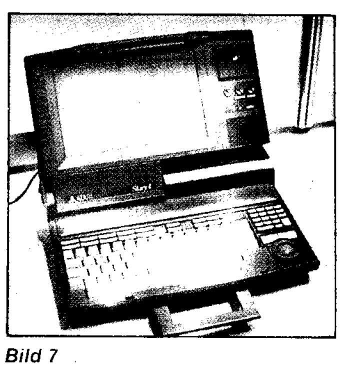


Bild 7

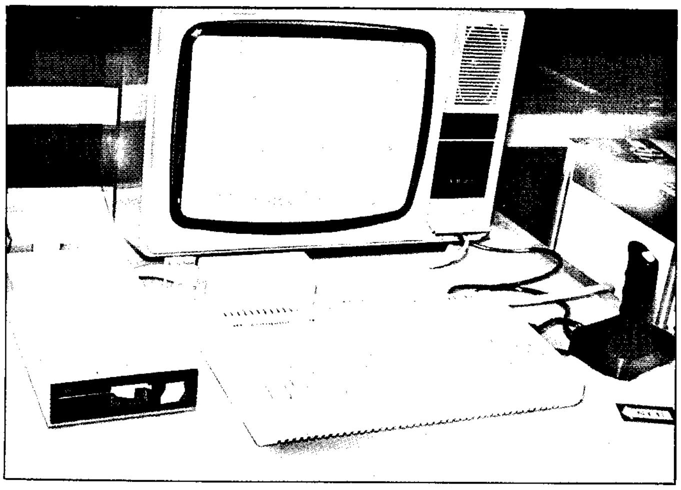


Bild 8

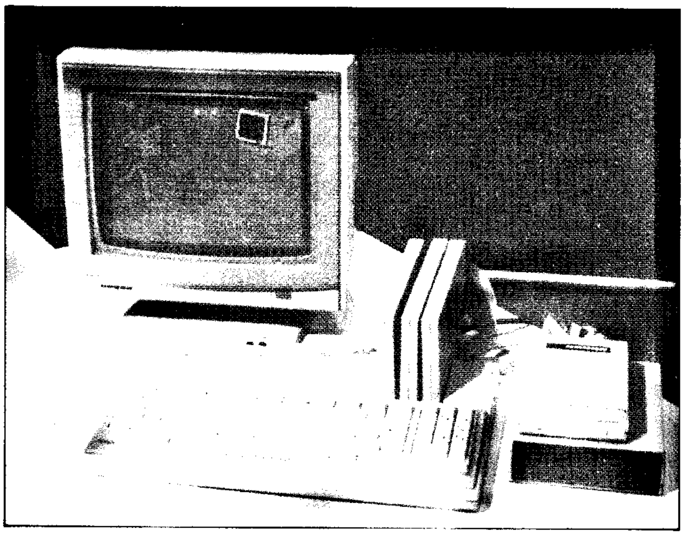


Bild 9

Neben den Anschlüssen für Kassettenrecorder, Joystick und Sound ist standardmäßig für Drucker eine Centronics-Schnittstelle vorhanden. Über ein sogenanntes Expansionsinterface kann auch – wie auf unserem Bild zu sehen – eine Diskettenstation angeschlossen werden.
Mehrere Heimcomputer wurden auch wieder im sowjetischen Pavillon ausgestellt. Der 16-Bit-Rechner **Poisk** (Bild 9) von der Kiewer Produktionsvereinigung Elektronmasch arbeitet mit dem 8088-kompatiblen Prozessor KM 1810 WM88, hat einen Hauptspeicher von 128 KByte RAM und 8 KByte ROM. Er soll eine Leistung von 1,25 MIPS erreichen. Mit den wahlweise anschließbaren Monitoren oder Fernsehgeräten können 320 × 200 Punkte in 4 aus 16 Farben oder 640 × 200 Punkte in 2 aus 16 Farben dargestellt werden. Als Speichermedium dient lediglich ein Kassettentonbandgerät.
Einen Heimcomputer mit einem 5¼-Zoll-Floppylaufwerk hatte das Pensaer Werk für Nachrichtenelektronische Geräte mit dem **Sura** zu bieten. Dafür verfügt er aber nur über einen 8-Bit-Prozessor und eine alphanumerische Bildschirmdarstellung. Der RAM hat eine Größe von 64 KByte und der ROM von 16 bis 32 KByte.

Der **Vektor-06C** von der Produktionsvereinigung Stschjotmasch aus Kischinjow arbeitet mit dem 8080-kompatiblen Prozessor KR 580 WM80A und soll damit immerhin 0,6 MIPS erreichen. Der RAM hat eine Größe von 64 KByte und der ROM von 512 Byte. Zum Speichern steht auch nur ein Kassettengerät zur Verfügung; der Anschluß eines externen Floppylaufwerkes ist möglich. Der Vektor-06C verfügt über eine Grafik mit 256 × 256 Punkten und 16 aus 255 Farben oder mit 512 × 256 Punkten mit 4 aus 255 Farben.

- **Peripherie**

Erstmals zur LFM vertreten war der bekannte Druckerhersteller Star Micronics, nach eigenen Angaben weltweit zweitgrößter Anbieter von Computerdruckern. Ähnlich anderen namhaften Produzenten reicht auch bei Star die Palette vom preiswerten Modell für den Hobbyanwender bis zu robusten Hochleistungsdruckern. Besonders für Kleinbetriebe dürfte die Farbversion des gutverkauften Einstiegsmodells LC 10 interessant sein. Mit dem Farbband, auf dem neben schwarz noch die Grundfarben rot, gelb und blau vorhanden sind, lassen sich durch Mischen insgesamt sechs Farben plus schwarz erzeugen. Die Farbauswahl erfolgt durch Steuerzeichen der jeweiligen Textverarbeitungssoftware. Der **LC 10 Color** (Bild 10) ist ein 9-Nadel-Drucker, der bei Schnellschrift bis zu 144 Zeichen/s und in Near Letter Quality bis zu 36 Zeichen/s (jeweils bei 12 cpi) auf das Papier bringt. Mit serienmäßiger Papier-Parkfunktion für Endlospapier, mit Schubtraktor, Papierwalze, halbautomatischem Papiereinzug, Trennautomatik für Endlospapier und vier eingebauten Schriftarten ist er zum Preis von etwa 700 DM ein sicher interessantes Angebot. Am oberen Ende der Leistungsskala stehen bei Star die Laserdrucker der Reihe LS-08 in den Versionen II, DB und DX. Allen gemeinsam ist die Druckgeschwindigkeit von 8 Seiten/Minute. Das Modell DB erlaubt gegenüber dem Modell II durch zwei Papierschächte das Drucken auf zwei getrennten Vorlagen; beispielsweise das erste Blatt einer Korrespondenz mit schon aufgedrucktem Firmenkopf und die Folgeseiten ohne diesen. Das Interessante an dem Modell **LS-08 DX** ist, daß hier gleichzeitig die Vorder- und Rückseite bedruckt werden kann, was eine erhebliche Zeitersparnis bei notwendigem Bedrucken der Rückseite bedeutet. Der Preis dieser Version liegt bei etwa 9500 DM. Zum Preis von etwa 3000 DM ist für alle Versionen auch eine PostScript-Option erhältlich.
Im Gegensatz zu Star Micronics präsentierte Seikosha bereits zum dritten Mal seine Produkte auf der LFM. Nennenswert hier vor allem die erstmals zur Systems 89 gezeigten neuen Drucker SP 2000, SL 92 und SL 532. Der **SP 2000** setzt die erfolgreiche 9-Nadeldrucker-SP-Serie fort und ist die Weiterentwicklung des SP 1600 (s. MP 8/89, S. 253). Gegenüber diesem erreicht der SP 2000 jetzt 40 Zeichen/s in Schönschriftdruck, er hat einen Schubtraktor, Papierparkfunktion, höheren Bedienkomfort und neben der parallelen Centronics-Schnittstelle erstmals serienmäßig gleichzeitig die serielle RS 232C-Schnittstelle. Der Preis liegt bei etwa 600 DM.
Auch der 24-Nadeldrucker **SL-92** (Bild 11) ist eine Weiterentwicklung, und zwar des SL-80. Gegenüber diesem ist der SL-92 schneller – 240 Zeichen/s in Schnellschrift und 80 Zeichen/s in NLQ Elite –, er besitzt auch einen Schubfaktor, einen vergrößerten Druckspeicher (44 KByte) und einen sogenannten Quiet-Modus, der das Druckgeräusch deutlich reduzieren soll. Preis: etwa 900 DM. Bei dem genannten 24-Nadler **SL-532** handelt es sich um einen Flachbettdrucker. Der Vorteil dieser Drucker – die seit kurzem von einigen Herstellern angeboten werden – liegt vor allem darin, daß das Papier nicht um die Druckwalze gerollt werden muß, sondern waagerecht durchgezogen wird. Damit können unflexible Materialien wie Pappe problemlos bedruckt werden; auch ist das Verwenden von bis zu vier Durchschlägen unproblematisch. Ausführlicher stellen wir dieses interessante Modell in Heft 8 vor.

Einen leistungsfähigen Tintenstrahlplotter bei Robotron mit dem Farbplotter **K 6421** an. Er soll als Ausgabegerät für 32-Bit-Rechner, wie den K 1840, dienen. Das zu verarbeitende oberflächenbehandelte Normalpapier kann eine Breite von A0 und eine Länge von 7 m haben. Die vier Grundfarben gelb, magenta, cyan und schwarz gestatten 64 Flächenfarben und 255 Farbmuster. Das verwendbare Normalpapier und die wasserlöslichen Tinten werden von Robotron als besonders umweltfreundlich hervorgehoben. Bei einer Übertragungsgeschwindigkeit von 450 KByte pro Sekunde wird eine Ausgabeleistung von 320 mm² pro Sekunde erreicht. Die Auflösung beträgt hierbei 5 Punkte pro mm bei einer Geschwindigkeit von ± 0,1 mm.

Eine interessante Produktpalette bot die Cherry Mikroschalter GmbH, die nicht nur durch ihr Firmenlogo – den Kirschenzweig – bekannt ist. Den

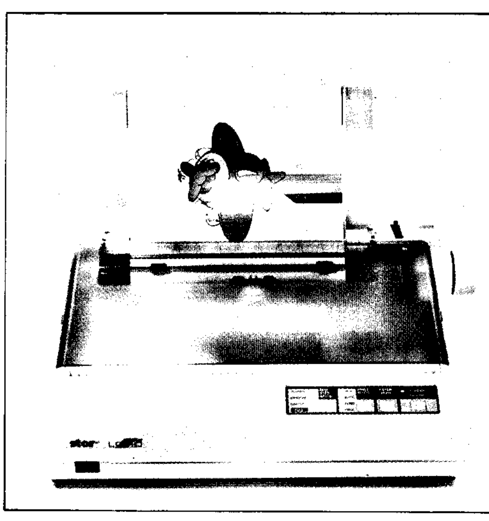


Bild 10

Bild 11

Schwerpunkt bilden bei Cherry die Tastaturen, von denen eine ganze Palette zu sehen war. Sie reicht von der Cherry 1000 – der Standardtastatur für PC/XT, /AT und PS/2 – über die Terminal-Tastatur Cherry 1300 bis hin zur multifunktionalen PC-Tastatur Cherry 2000 mit Magnetkartenleser, Barcodeleser, Maus und Display. Für viele PC-Anwender dürfte die **Cherry 2100** von großen Interesse sein. Sie bietet neben dem gewohnten Leistungsumfang einer PC-Tastatur 23 zusätzliche Funktionstasten, die individuell programmiert werden können (im Bild 12 die oberen zwei Reihen). Diese Funktionstasten lassen sich in 10 Ebenen mit Hilfe der (24sten) Cherry-Taste belegen, wobei die Programmierung in Online-Betrieb erfolgen kann, also die Speicherung der Zeichen während der Eingabe möglich ist. Die Speicherkapazität beträgt zirka 2 KByte (optional 8 KByte) bei variabler Aufteilung. Der Programmiermodus kann über DIL-Schalter gesperrt werden. Die Ausgabegeschwindigkeit der Zeichen läßt sich einstellen. Sämtliche Funktionstasten sind mit „relegendable" Tastenköpfen ausgestattet, welche eine individuelle Beschriftung zulassen. Sämtliche eingestellten Parameter werden bei der Cherry 2100 gespeichert und bleiben auch nach Abschalten des Systems für typisch 10 Jahre erhalten.

Der Haupteinsatzbereich der Cherry 2100 ist das Gebiet der Textverarbeitung. Das Keyboard ist mit MX-Tastmodulen mit einem Betätigungshub von 4 mm und mit „Gold-Crosspoint"-Kontakten bestückt. Die Lebensdauer der Einzeltaste ist $> 5 \times 10^7$ Betätigungen (bei einer Zuverlässigkeit von: MCBF = 10^9 Betätigungen, MTBF = 80 000 Stunden.

Weitere Produkte von Cherry waren Mikrotaster, Elektrolumineszenz- und Plasmadisplays sowie das **Cherry Grafik-Tablett Mk. 4**. Dieses Grafik-Tablett ist geeignet für diverse Anwendungssoftware. Es stehen eine serielle RS 232 und eine TTL-Schnittstelle zur Verfügung. Zusätzlich bietet das Mk. 4 wählbare Schnittstellenparameter in bezug auf Datenformat, Parität, Stopbits, Auflösung, Übertragungsraten und Maus-Emulation. Die Eingabe erfolgt entweder über Cursor oder über Griffel. Der Cursor verfügt über vier Eingabetasten mit Druckpunkt und ein Fadenkreuz. Eine der verschiedenfarbigen Tasten dient der Dateneingabe, während die drei anderen frei programmiert werden können. Der Griffel eignet sich besonders zur Menüauswahl und für Freihand-Skizzen. Ergonomisch geformt ist er mit einem Druckpunkt-Schalter und einer komfortablen Rollerball-Spitze ausgestattet. Durch eine automatische Erkennung der genutzten Einheit können Griffel und Cursor auch gleichzeitig angeschlossen werden. Gestaltet ist das Cherry Mk. 4 in ergonomisch zweckmäßigem Flachbett-Design, wodurch die kompakte Arbeitsfläche im Format DIN A3 mit dem Gehäuse bündig ist und somit einfachere Griffel- und Cursorbewegungen erlaubt. Unter einer austauschbaren, transparenten Folie können schnell und sicher größere Dokumente oder Menüs plaziert werden. Klebestreifen zur Fixierung werden dadurch entbehrlich.

- **Bauelemente**

Die Leipziger Messe zählte noch nie zu den ausgesprochenen Bauelemente-Messen. Während ausländische Firmen nur selten ein paar ausgewählte Schaltkreise für die Mikroelektronik anboten, zeigten die DDR-Betriebe dagegen ihre gesamte Produktpalette. Zur diesjährigen Frühjahrsmesse waren aber auch diese Betriebe zurückhaltender. Das ist sicherlich der wirtschaftlichen Entwicklung geschuldet, die die Produktion von kostenungünstigen Schaltkreisen nicht mehr als sinnvoll erscheinen läßt. Trotzdem konnten wir einige Highlights der DDR-Industrie ausfindig machen.

Da wäre beispielsweise der digitale Signalprozessor **U 320C20** vom Forschungszentrum Mikroelektronik Dresden (ZMD) zu nennen (siehe Bild 13). Er ist der erste Signalprozessor aus DDR-Produktion. Der zum TMS32020 von Texas Instruments funktionskompatible Prozessor wurde jedoch im Unterschied zu diesem in der CMOS-Technologie CSGT4 ausgeführt. Er besitzt eine Daten- und Adreßwortbreite von 16 Bit und einen On-chip-RAM von 544 Wörtern. Seine Daten- und Programmadreßräume betragen je 64 K Wörter. Die Arithmetik- und Logikeinheit und der Akkumulator verfügen über eine Wortbreite von 32 Bit. Der U 320C20 arbeitet mit 109 Befehlen, zu denen auch Befehle zur Unterstützung von Gleitkommaoperationen gehören. Bei einer Taktfrequenz von 20 MHz hat er eine Leistungsaufnahme von 0,5 W. Seine Blocktransportrate beträgt 5 Mio Wörter pro Sekunde. Er befindet sich in einem QFP 68-Gehäuse und ist daher nicht pinkompatibel zum PGA 68-Gehäuse des TMS32020.

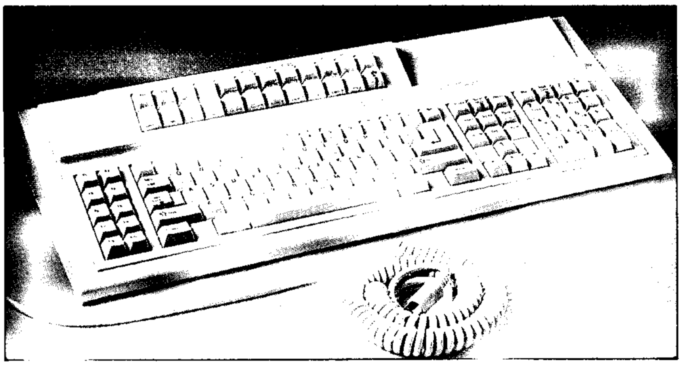


Bild 12

Erstmals auf der Leipziger Messe zu sehen war auch der erste 32-Bit-Mikroprozessor aus DDR-Produktion, der **U 80701** vom Kombinat Mikroelektronik Erfurt. Allerdings wurde von dem Prozessorsystem U 80700 nur die CPU 80701 ausgestellt. Auch der von Robotron angekündigte Rechner K 1820, der auf der Basis dieses Prozessorsystems entwickelt wurde, war, wie schon erwähnt, in Leipzig nicht vertreten (vgl. auch MP 11/89, Seite 322).

Bessere Aussichten hat dagegen das 16-Bit-Mikroprozessorsystem U 80600, das im AT-kompatiblen EC 1835 von Robotron eingesetzt wird. Die ersten Schaltkreise dieses Systems zeigte das Kombinat Mikro-

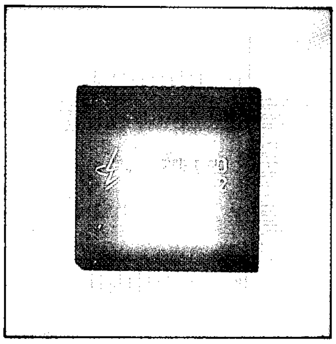


Bild 13

elektronik bereits zur LFM '89 (MP stellte diese Schaltkreise im Heft 5/89 vor). Zur diesjährigen Frühjahrsmesse waren der Harddisk-Controller U 82062 und der Taktgenerator DS 80612 erstmals zu sehen. Der **U 82062 DC05** (Bild 14) setzt 8-Bit-Paralleldaten in einen MFM-kodierten seriellen Datenstrom mit einer Übertragungsrate von 5 MBit pro Sekunde um. Er benötigt für seine Arbeit außerdem extern eine Taktseparatorschaltung mit einer PLL für Datenleseoperationen und eine Präkompensationsschaltung. Die komplette Baugruppe ermöglicht den Datenaustausch entsprechend den Schnittstellenstandards ST 506/ST 412 von Seagate und SA 1000 von Shugart. Mit dem U 82062 DC05 können vier Festplattenlaufwerke angesteuert werden. Der Chip befindet sich in einem 40poligen Plastgehäuse; er kann maximal mit nur 5 MHz getaktet werden.

Im Unterschied zu den anderen bisher vorgestellten Schaltkreisen des System U 80600, die in MOS-Technologie hergestellt werden, ist der **DS 80612** ein Schottky-TTL-Schaltkreis. Der DS 80612 DC erzeugt Takt-, Ready- und Reset-Signale für 16-Bit-Prozessorsysteme. Er besitzt einen quarzgesteuerten Oszillator, einen Taktgenerator mit MOS-kompatiblen Ausgängen, die Ready-Synchronisation und die System-Reset-Generie-

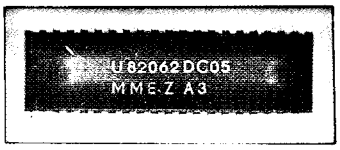


Bild 14

rung. Sein Chip befindet sich in einem 18poligen Plastgehäuse in DIP-Bauform; Angaben zur Taktfrequenz wurden nicht gemacht.

Das Gate-Array-System **U 5300** vom Forschungszentrum Mikroelektronik Dresden – im vorigen Jahr noch in der Zeiss-Messehalle zu finden – war in diesem Jahr am Stand des Kombinates Mikroelektronik ausgestellt. Die Palette des Systems U 5300 hat sich inzwischen vergrößert. Zu den Mastern U 5301 und U 5302 gesellten sich der U 5303, der U 5304 und der U 5405. Wesentliche Unterschiede zwischen den verschiedenen Mastern sind die Zahl der verfügbaren Logikelemente und damit die Chipgröße und die Pinzahl der Bauelemente sowie die mögliche Taktfrequenz (siehe hierzu Tafel 1). Der U 5302 läßt als einziger die Integration von 8 Speicherblöcken zu je 64 × 9 Bit (RAM) oder 128 × 8 Bit (ROM) zu. Allen gemeinsam ist die Gatterverzögerungszeit von 0,9 ns und das Taktsystem mit zwei unabhängigen Takten. Alle Pins können CMOS- und TTL-kompatibel generiert werden. Die Chips sind mit QFP-Gehäusen (flat pack) verkappt. Für die Entwicklung dieser Kundenwunsch-Schaltkreise bietet ZMD mehrere Varianten vom Verkauf des CAD-Systems an den Kunden über Leasing des CAD-Systems bis hin zur Auftragsentwicklung durch den Hersteller an.

Anwendungen und Software

- **CAD/CAM**

Am Stand des West-Berliner Handelshauses Transcommerz wurde mit dem vom Leitzentrum für Anwendungsforschung gezeigten Leiterplattenentwurfssystem PCLES eine modulare CAE/CAD-Lösung vorgestellt, die alle Prozesse von der Schaltungsentwicklung bis zur Ausgabe von NC-Steuerdaten umfaßt. Neben dem universellen Bauelemente-Datenbanksystem CADBEK (ca. 20 000 Einträge), dem Symbol- und Bauelementekatalog, beides vom ZFTN des Kombinates Nachrichtenelektronik, gehören dazu das CAD-System CADSTAR, mehrere Postprozessoren und das Artworksystem ARTY für Technologen und Fertigungszentren. Das LfA Berlin tritt demnächst als autorisierter Händler für CADSTAR 5 auf, das neueste Update der PC-Elektronik-Design-Software der englischen Firma Racal-Redac. Es ist ein Komplettsystem für die Elektronikentwicklung, bestehend aus Stromlaufplanpaket (SCM) und dem Leiterplatten-CAD-Paket (PCB). Beide Appli-

Tafel 1 Gate-Array-System U 5300

Master	U 5301	U 5302	U 5303	U 5304	U 5305
Pinzahl	124	124	68	84	52
Gatteräquivalente	8160	5040	2640	4056	1176
-Flipflops	360	200	120	182	70
-Logikgatter	3848	2640	1200	1872	672
Taktfrequenz (MHz)	>40	>30	>60	>60	>80

kationen verfügen über eine Vielfalt von leistungsstarken automatischen und interaktiven Entwurfshilfen. CADSTAR SCM unterstützt das Erstellen hierarchischer Schaltungsentwürfe. Es enthält außerdem ein Katalog-Management und einen Postprozeß, der die Ausgabe von Testplots oder von Bauteil- und Netzlisten für die Simulation (CADAT-Logiksimulator) und PCB ermöglicht. CADSTAR PCB hat volle SMC- und Mehrlagenfähigkeiten (maximal 16 Lagen) und bewältigt bis zu 5400 Verbindungen und 1023 Bauelemente pro Leiterkarte. Diese können auf einer Fläche von bis zu 32 Zoll × 32 Zoll bei einer maximalen Auflösung von 1/1000 Zoll angeordnet werden. Die Layoutdaten von CADSTAR können für Postscript-Drucker, NC-Bohrmaschinen, Pen- und Photoplotter konvertiert werden. Mit dem Artworksystem können u. a. die Gerberdateien aus dem PCB grafisch editiert und Excellon-Bohrfiles und Nutzen erstellt werden. Gezeigt wurde das CAD-System auf einem AT 386SX von Tulip, der mit einem CONRAC-Monitor ausgerüstet war. Gekoppelt war dieser Rechner über ein Novell-Netz u. a. mit dem rechnergestützten Organisations-, Sekretariats- und Informationssystem ROSI, ebenfalls vom LfA Berlin.

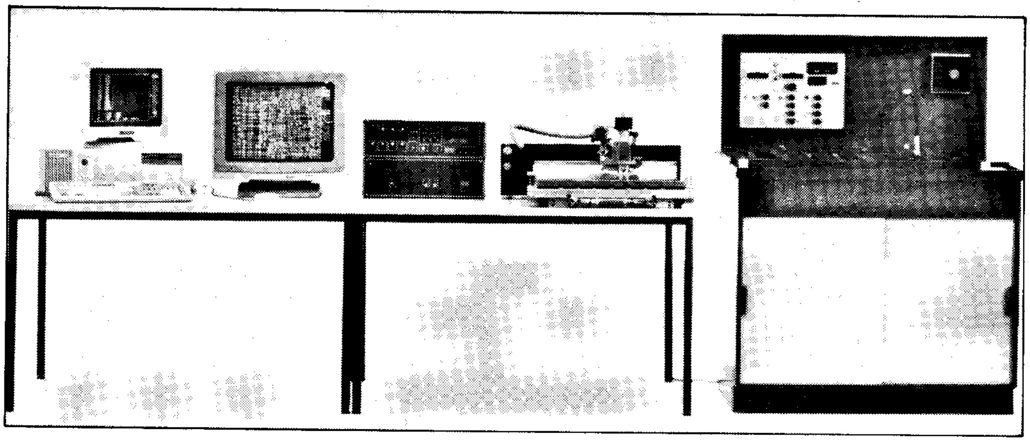


Bild 15

Bild 16▼

Das von der AUCOTEC GmbH Hannover vorgestellte CAD/CAE-System **ELCAD** dient insbesondere dem Erstellen und Auswerten sowie der Dokumentation von Schalt- und Steuerungsplänen in den Bereichen Elektrotechnik, Elektronik, Hydraulik und Pneumatik. Sowohl unter MS-DOS als auch unter Unix kann ELCAD genutzt werden; der Einsatz ist auch im Netz möglich und Daten – insbesondere Grafikdaten – anderer Software können eingelesen und weiterverarbeitet werden. Entsprechend den speziellen Hard- und Softwarevoraussetzungen wird ELCAD kundenspezifisch konfiguriert. Die vom System erzeugten Daten sind dabei unabhängig von den konkreten Bedingungen (Hardware, Betriebssystem) und können auch auf anderer Hardware unter dem jeweils anderen Betriebssystem weiterverarbeitet werden. ELCAD wurde unter anderem für die Anwendung im Maschinen-, Anlagen- und Fahrzeugbau im Signal- und Schaltanlagenbau und in der Meß- und Regelungstechnik, aber auch der Betriebsinstandhaltung entwickelt. Nach sehr kurzer Zeit soll dieses System auch ohne spezielle Computer- oder CAD-Kenntnisse erfolgreich eingesetzt werden können. Dem sollen auch die Bedienerführung – wahlweise in deutscher, englischer, französischer oder schwedischer Sprache –, ein klar gegliedertes Handbuch und die von AUCOTEC angebotenen, ergänzenden Schulungen dienen. Die Programmanweisungen werden mit einem Zeichenstift über ein Grafiktablett eingegeben, und so soll der Konstrukteur seine gewohnte Arbeitsweise größtenteils beibehalten können. Sind Detailänderungen an einer Zeichnung erforderlich, werden automatisch alle abgeleiteten Komponenten aktualisiert.
Die Grundlagen des Systems sind die Normen nach DIN; es sind aber weitere Normen lieferbar (z. B. US, ISO, Hausnormen der Automobilindustrie) und auch das Anlegen von beliebigen weiteren Normbibliotheken ist möglich. Damit können die Konstrukteure in der ihnen vertrauten Umgebung arbeiten. Das fertige Projekt soll dann von ELCAD automatisch in jede andere Norm übertragen werden können. Ebenso sollen auch die Dokumentationen neben DIN nach jeder anderen Werknorm möglich sein. Für einige Bereiche (z. B. Elektrotechnik und Elektronik) liegen umfangreiche Symbolbibliotheken vor, die vom Anwender zudem noch beliebig erweitert werden können. Der Programm-Modul **ELMAT** erstellt auf der Grundlage einer vom Anwender selbst gestalteten Datenbank simultan eine Liste aller in den Zeichnungen verwendeten Bauteile. Die freie Gestaltung der Liste gestattet beispielsweise die Ausgabe von Rohstücklisten, Material-, Montage- oder Bestellisten, die neben den technischen Daten auch Angaben zu den Preisen oder zum Platzbedarf enthalten können.
In der Leiterplattenentwicklung, insbesondere mit SMD-Bauelementen, ist die Prototypenherstellung oft langwierig und kostspielig. Projekte werden durch die Abhängigkeit vom Servicebetrieb bzw. von der Serviceabteilung verzögert. Eine eigene Herstellung in der herkömmlichen Ätztechnik scheidet oft aus, nicht nur wegen der Entsorgungsprobleme, sondern auch wegen der immer höher werdenden Bestückungsdichte. Hinzu kommen gestiegene Anforderungen beim präzisen CNC-Bohren. Eine Alternative bietet die Komplettlösung **COLOR CAM** (Bild 15) von der Hannoveraner Firma LPKF CAD/CAM SYSTEME GmbH. Das COLOR CAM besteht aus einem AT-kompatiblen PC, dem LPKF Fräsbohrplotter und der Durchkontaktierungsanlage LPKF CONTAC (im Bild v. l. n. r.). Der PC kann einen 80286 oder einen 80386 enthalten und wird zusätzlich mit einer T414-Transputerkarte mit 4 MByte RAM on board und einer Leistung von 10 MIPS bestückt. Die zusätzliche ACRTC-Grafikkarte mit schneller Bildmanipulation ermöglicht eine Auflösung von 1024 × 768 Bildpunkten. Rechner- und Grafikmonitor sind getrennt. Für die Ausgabe können beispielsweise Plotter, Fotoplotter oder CNC-Systeme über alle üblichen Softwareschnittstellen angeschlossen werden. Ein vielfältiges Softwareangebot unterstützt den Schaltplanentwurf, die Entflechtung sowohl interaktiv mit Dialog-Autorouter als auch automatisch mit LPKF-RE-ROUTER sowie eine professionelle Bibliothekstechnik.
Für die Herstellung der Leiterplattenprototyps werden die Layoutdaten mit dem Programm ISOLATE invertiert und dann der Fräsbohrsteuerung übergeben. Das Isolieren erfolgt durch Umfräsen der Leiterbahnen mit einem Gravierabstand von 0,2 bis 0,4 mm; aber auch das herausfräsen von Flächen zwischen den Bahnen ist möglich. Zur Durchkontaktierung mehrlagiger Platinen direkt im Entwicklungslabor dient das 4-Kammer-System LPKF CONTAC. Hervorzuheben ist hierbei, daß es keine Entsorgungsprobleme der chemischen Bäder gibt. Bei Nachlassen der Badqualität kann die Flüssigkeit zum Erneuern an LPKF geschickt werden. Ebenfalls interessant für Leiterplattenentwickler ist der **DIGIMASTER** von LPKF. Er stellt ein komplettes EDV-System zum Scannen und digitalisieren jedweder Vorlagen wie Filme, Zeichnungen, Skizzen und Layouts dar. Mit diesem System können auch alte Layoutvorlagen restauriert werden, wenn Filme oder Datenträger nicht mehr vorhanden sind. Die Vorlagen werden über einen Scanner in den PC eingelesen und dort im Gerberformat abgespeichert. Bei der Überarbeitung können Zeichnungsteile weggelassen oder hinzugefügt werden. Das Programm macht dies durch unterschiedliche Farbdarstellung auf dem Bildschirm deutlich.

Für alle Betriebe, die in der nahen oder auch ferneren Zukunft computerintegrierte Produktionsabschnitte aufbauen wollen, bietet die Universität Dortmund mit dem **CIM-Lernbetrieb** (Bild 16) eine Möglichkeit, sich rechtzeitig auf diese Anforderungen einzustellen. Ein funktionstüchtiger, vernetzt gesteuerter Modellbetrieb mit Konstruktion, Materialtransport, Fertigung, Montage, Auftragsverwaltung, Produktionsüberwachung einschließlich der Qualitätskontrolle kann sowohl zur Schulung von Planern und Entscheidern als auch von Facharbeitern und Anlagenführern genutzt werden. Somit ist es nicht erforderlich, komplexe und hochproduktive Industrieanlagen für Schulungszwecke zu nutzen und dabei das Risiko von schwerwiegenden Havarien eingehen zu müssen. Die erforderlichen Schulungsmaßnahmen für den vorgesehenen Personenkreis können damit rechtzeitig vor der Einführung der neuen Produktion abgeschlossen sein.

Am Stand von Atari stellte die rhothron gmbh, Homburg, eine interessante Atari ST-Anwendung aus. Die Produktlinie **rho-prof** (Bild 17) faßt leistungsstarke Hard- und Softwarekomponenten zu einem modularen Komplettsystem für Meßwerterfassung und -auswertung, für Prozeßüberwachungen, Steuerungen und Regelungen sowie für Qualitätssicherung und Dokumentation zusammen. Zur Ausstattung gehört der multiprozessorfähige Masterrechner mit 68000-CPU, 1 bis 4 MByte RAM, eine parallele und eine serielle Schnittstelle, ein 3,25-Zoll-Floppylaufwerk mit 720 KByte, eine akkugepufferte Echtzeituhr, eine Harddisk mit 60 bis 160 MByte und optional eine 5,25-Zoll-Floppy. Untergebracht ist alles in einem Tischgehäuse mit 9 bis 20 Steckplätzen (je nach Gehäusegröße) für VEM-Baugruppen. Die freien Steckplätze können mit diversen Zusatzkarten für Meßwertaufnahme oder für Netzwerkarbeit belegt werden. Als Monitor dient ein 12-Zoll-Monochrommonitor (papierweiß) mit einer Auflösung von 640 × 400 Punkten. Für die Verwendung von PC-Software kann der rho-prof zusätzlich mit einem V30-Prozessor

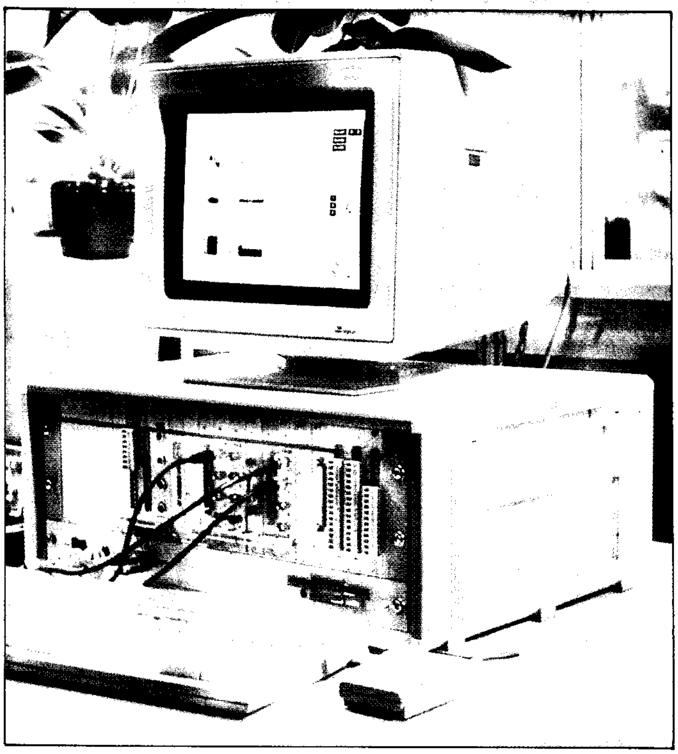


Bild 17

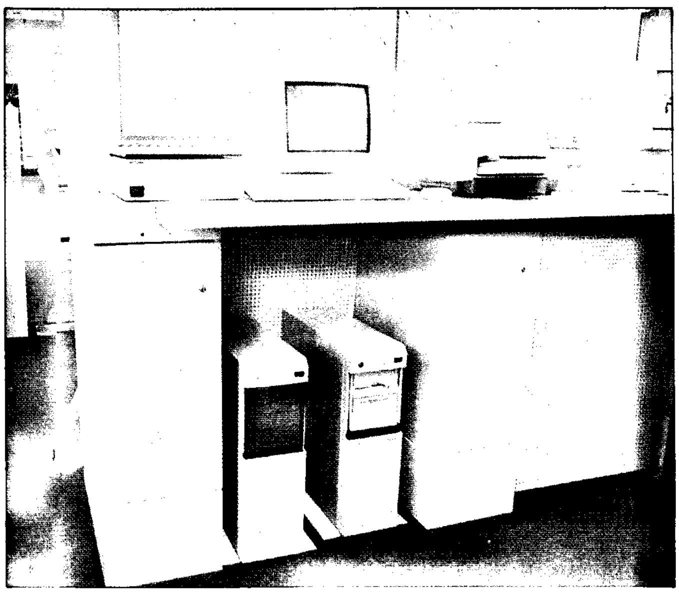


Bild 18

von NEC **bestückt werden. Software**mäßig kann zwischen beiden Prozessoren umgeschaltet werden. Festplatten- und Diskettenlaufwerke können dabei Dank der identischen Formate gemeinsam genutzt werden.

• **Desktop Publishing**

Wer sich auf der LFM nach Möglichkeiten zum Desktop Publishing, also dem „Publizieren am Schreibtisch" umsah, konnte unter verschiedenen Möglichkeiten wählen. Am Siemens-Stand war ein DTP-System zusammengestellt, dessen Hardwarekomponenten alle aus dem gleichen Hause stammten – was durchaus Vorteile bieten kann. Kern war der **PCD-3T**, ein PC im Towergehäuse (Bild 18) mit dem 32-Bit-Prozessor Intel 80386 (20 MHz, umschaltbar auf 6 oder 8 MHz) und 4 MByte Hauptspeicher. Mit Ausnahme des Videocontrollers befindet sich die komplette Systemhardware auf der CPU-Karte. In der Grundkonfiguration enthält der PCD-3T ein 3,5-Zoll-Diskettenlaufwerk (1,44 MByte) und eine Festplatte mit 70, 155 oder 260 MByte. Als Eingabegerät war der Scanner HighScan 400 angeschlossen – Auflösung 400 dpi, bis 64 Graustufen, Eingabedauer 11 Sekunden –, als Ausgabegerät der Laserdrucker HighPrint PT 10. Als Software wurde das bekannte DTP-Programm Pagemaker verwendet.

Ein weiteres DTP-Paket – für Atari ST-Rechner – stellen wir Ihnen in diesem Heft auf der 2. Umschlagseite etwas ausführlicher vor.

• **Standardsoftware**

Da die diesjährige Leipziger Frühjahrsmesse von einer ungewohnten Weltoffenheit gekennzeichnet war, erklärt das auch ein doch deutlich gestiegenes Interesse großer Firmen an dieser Präsentation. Erstmals in Leipzig vertreten waren solche Softwarefirmen wie die Oracle Deutschland GmbH und die C.A. Computer Associates GmbH, letztere sogar im März schon mit konkret abgeschlossenen Kooperationsverträgen mit dem Leitbetrieb für Anwendungsforschung (LfA) zum Vertrieb der Produkte in der DDR (vgl. auch MP 5/90, S. 105).

Nun jedoch zu den Produkten im einzelnen. Beginnen wir gleich mit Computer Associates, dessen bei PC-Nutzern bekanntestes Produkt wohl das bereits auf den CP/M-Rechnern eingesetzte Kalkulationsprogramm Supercalc sein dürfte. In Leipzig nun wurde den Interessenten die aktuelle Version Supercalc 5 vorgestellt. Und obwohl etwas versteckt am Gemeinschaftsstand der USA, soll das Interesse an Produkten von C.A. doch recht groß gewesen sein.

Supercalc 5 ist ein Tabellenkalkulationsprogramm für IBM-kompatible PCs mit mindestens 512 KByte RAM und zwei Diskettenlaufwerken ab 360 KByte Speicherkapazität. Günstiger ist allerdings eine Festplatte, um die enthaltenen Möglichkeiten ausnutzen zu können. Eine Tabelle kann bis zu 256 Seiten umfassen, 256 Tabellen (oder neudeutsch: Spreadsheets) können im Speicher oder auf der Platte miteinander verbunden werden, wobei jeweils drei Tabellen gleichzeitig auf dem Bildschirm angezeigt und bearbeitet werden können (Bild 19). Bei der Arbeit im Netz, die ja in Zukunft immer mehr an Bedeutung gewinnen wird, können beliebig viele Nutzer gleichzeitig beliebig viele Dateien öffnen (und damit für andere Bearbeitungen sperren), anzeigen und bearbeiten. Es ist dabei sowohl möglich, eine Verbindung zu Großrechnern herzustellen als auch Lotus 1-2-3-Dateien zu bearbeiten. Die vollständige Kompatibilität zu Lotus-Dateien, -Menüs und -Makros gestattet das direkte Lesen und Schreiben von Lotus-Dateien und auch das unproblematische Verbinden mit Supercalc 5-Dateien. Sowohl im Direktmodus als auch in Programmen können Lotus- und Supercalc-Befehle gemischt benutzt werden. Supercalc 5 unterstützt LIM Version 4.0 und kann somit 32 MByte Hauptspeicher direkt nutzen. Des weiteren enthält Supercalc 5 einen kleinen Texteditor für Mitteilungen und Kommentare und kann um ein Datenbanksystem erweitert werden. Da es möglich ist, auch umfangreiche Anwendungen im Netz zu programmieren, fällt angenehm auf, daß ein integrierter Debugger – eine aus vielen Programmiersprachen her bekannte Hilfe für den Programmierer – das Aufspüren der oftmals doch sehr versteckten logischen Fehler unterstützt (siehe auch das Titelbild dieses Heftes). Neben der Analyse der Makros einschließlich ihres Ablaufs können Haltepunkte gesetzt und so die Befehlsketten der Makros schrittweise ausgeführt werden. Ebenso hilfreich ist das Markieren und Anzeigen aller Zellen (Felder), die sich seit der letzten Berechnung geändert haben und ein Report über die Ausführung des Makros. Die gewonnenen Daten können als qualitativ hochwertige Präsentationsgrafiken sowohl zwei- als auch dreidimensional auf Bildschirm oder Drucker ausgegeben werden. Aus über 100 Diagrammtypen können Sie die günstigste Darstellungsform für die von Ihnen beabsichtigte Aussage und zur Beschriftung aus mehreren Schriftarten auswählen. Auf Laserdruckern bzw. Postscript-Geräten können somit hochwertige Reports erzeugt werden (Bild 20).

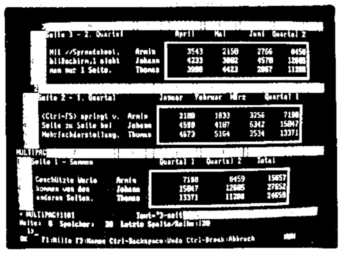


Bild 19

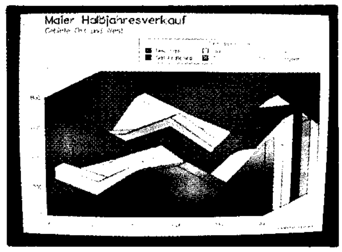


Bild 20

Warum gerade dBase als Datenbanksystem einen so großen Erfolg hatte und auch immer noch hat, ist sicher auch angesichts zahlreicher anderer Systeme nicht völlig verständlich. Ein Hauptgrund wird aber sein, daß dBase seit den altvorderen CP/M-Zeiten mit dem Computeranwender mitgewachsen ist, und somit von der Macht der Gewohnheit profitieren konnte. Durch die ständig steigende Leistungsfähigkeit der PCs was einerseits die höheren Kapazitäten der Massenspeicher und andererseits die höheren Verarbeitungsgeschwindigkeiten betrifft, kommt jetzt auf diesem Gebiet eine immer größere Konkurrenz aus dem Bereich der bisherigen Groß- und Minirechner. **SuperDB** und die Weiterentwicklung **SuperDB2** sind dabei Produkte, die durchaus als interessante Datenbanksysteme bezeichnet werden können. Während SuperDB2 einen IBM-kompatiblen PC XT/AT oder das PS/2 erfordern, mindestens 512 KByte Hauptspeicher und eine Festplatte und ein Diskettenlaufwerk bzw. zwei Diskettenlaufwerke sowie CGA-, EGA-, VGA oder Hercules-Grafikkarten voraussetzen, soll SuperDB bereits auf einem EC 1834 der Firma Robotron laufen. Die wesentlichen Neuerungen in SuperDB2 sind in einer höheren Arbeitsgeschwindigkeit und in einer höheren Anzahl möglicher Verknüpfungen zwischen den Dateien – auch bei Reports – zu sehen. Beide Systeme sind durch eine einheitliche und leicht zu bedienende Nutzeroberfläche gekennzeichnet und ermöglichen es, automatisch Daten auszuwerten und die ausgewerteten Daten in Listen oder grafisch darzustellen.

Kenntnisse des Datenbankmanagements sollen ebensowenig wie Programmierkenntnisse erforderlich sein, um Datenbanken anlegen, bearbeiten und auswerten zu können. Aber auch bereits bestehende Datenbanken können in den meisten Fällen übernommen oder bearbeitet, das heißt auch wieder geschrieben, wer-

den; so beispielsweise für die Formate ASCII, dBase, Word und Wordperfect. Mehrere statistische Berechnungen wie Standardabweichung und Varianz können Sie vornehmen und die Daten in Kreis-, Balken-, Linien- oder Schachteldiagrammen leicht zugänglich machen. Den Zugriff auf die direkten Daten allerdings können Sie gänzlich unterbinden oder aber über ein Paßwort das Lesen von Daten gestatten. Ebenso kann das Ändern der Daten und der Dateistruktur nur von dem dazu berechtigten Personenkreis vorgenommenwerden. Einige weitere technische Daten in Zahlen finden Sie in Tafel 2. Als ein drittes Softwareprodukt von C.A. Computer Associates möchten wir besonders den zukünftigen Wirtschaftsmanagern unter Ihnen **Superprojekt Plus** vorstellen, ein Paket für das Projektmanagement auf IBM-kompatiblen PCs (XT/AT) unter MS-DOS 2.0 oder höheren Versionen. Weitere Hardwarevoraussetzungen sind eine Festplatte oder zwei Diskettenlaufwerke und mindestens 360 KByte freier Hauptspeicher. Netzpläne und Balkendiagramme der Ressourcen verdeutlichen die einzelnen Phasen sowie die erforderlichen zeitlichen, personellen und materiellen Voraussetzungen eines Projektes. Die kontextbezogenen Hilfen, flexible Editierfunktionen, das Drucken individuell zugeschnittener Berichte und die Übernahme/Übergabe von Daten anderer Software sollen einerseits eine kurze Einarbeitungszeit garantieren und andererseits eine hohe Akzeptanz des Pakets sichern. Der Basiszeitplan, die geplante und die tatsächliche Aufgabendauer können auch für mehrere Haupt- und Unterprojekte angegeben, verglichen und zusammengefaßt werden, und für jede Aufgabe und jedes Projekt kann eine Stichterminplanung vorgenommen werden. Die gesamten Kosten (Fixkosten, variable Kosten, und auch die Gegenüberstellung der geplanten und der tatsächlichen Gesamtkosten) können für jedes Projekt und jede Aufgabe ebenso wie die einmaligen Fixkosten und die variablen Kosten für jede Ressource über den Zeitraum verteilt, zu Beginn oder zum Ende des Projektzeitraumes entstehend, geplant werden. Minderauslastungen und Überlastungen von Ressourcen können nicht nur für mehrere Projekte angezeigt werden, sondern die Lösung von Konflikten auch zwischen mehreren Projekten wird durch das Softwarepaket unterstützt.

Die Oracle Deutschland GmbH, eine Tochter der amerikanischen Oracle Corporation, arbeitet seit April 1990 auch mit Robotron Projekt Dresden zusammen und bietet Softwarepakete für das Informationsmanagement, für eine produktive Anwendungsentwicklung und für herstellerunabhängige Systemintegration unterschiedlicher Datenbanksysteme, die auf der relationalen Technologie basieren und sowohl auf Großrechnern als auch auf PCs laufen. Damit ist Oracle-Software ein weiteres Beispiel dafür, wie die in der letzten Zeit stark gestiegene Leistungsfähigkeit der PCs den Einsatz der bislang nur auf Großrechnern lauffähigen Pakete von dem oberen Leistungsniveau in diesen Bereich vordringen und damit herkömmlichen Programmen Konkurrenz bieten. Die Zukunft moderner Informationsverarbeitung liege, so Oracle selbst über ihr Konzept, nicht in immer gigantischeren Großrechnern und der zentralen Versorgung von immer mehr Arbeitsplätzen, sondern in einem Netz aus einem Host (Wirt) als zentralem Steuerrechner und verteilten Bereichs- und Arbeitsplatzrechnern wie Grafikworkstations oder PCs. Das Netz verbindet also verschiedene Datenbanksysteme miteinander, wobei der Nutzer auf diese Daten zugreifen kann, ohne daß ihm der Speicherort bekannt sein muß. Mit **SQL*Star** will Oracle nun eine wirtschaftliche Basis für eine verteilte Datenbankarchitektur, für eine herstellerunabhängige Dezentralisierung und ein einheitliches Informationsmanagement sowohl für Groß- und Minirechner als auch für PCs bieten. Eine verteilte Datenbank unter SQL*Star besteht aus einem Netz lokaler Datenbanken, die durchaus auch auf unterschiedlichen Systemen liegen können, sich aber für den Benutzer als ein einziges System darstellen. Als ein offenes System kann aus SQL*Star auf Daten fremder Datenbanken zugegriffen werden, auf die volle Kompatibilität zum relationalen Datenbanksystem DB 2 von IBM wird jedoch besonderer Wert gelegt. Die Standardabfragesprache SQL für Datenbanken (Structured Query Language) bildet dabei die Grundlage für den Zugriff auf unterschiedlichste Datenbanken. Mehrere Tools erweitern die Möglichkeiten der komfortablen Nutzung von Oracle-Datenbanken, so SQL*Forms, SQL*Plus, SQL*Reportwriter, SQL*Calc (ähnlich Lotus 1-2-3), SQL*Net und SQL*Connect, (ein Zusatzpaket, das das Lesen und Schreiben von Daten von Fremddateien ermöglicht). SQL*Forms (Bild 21) generiert komplexe Bildschirmanwendungen, soll die konventionelle Programmierung überflüssig machen und bis zu achtzig Prozent bisheriger Programmierzeit einsparen. Das lauffähige Anwendungsprogramm wird aus der logischen Struktur der Daten und aus ihrem Format in den Masken generiert. Das zusätzliche Einbinden individueller Routinen in konventionellen Programmiersprachen wie Pascal und C, aber auch Fortran, Cobol, PL/1 oder Ada ist möglich.

SQL*Plus bietet neben dem vollen SQL-Sprachumfang leistungsfähige Erweiterungsfunktionen, beispielsweise für den Datentransfer. Ebenfalls enthalten sind Report- und Druckoptionen, mathematische und logische Opratoren. Der Einsatz von Variablen und die gleichzeitige Arbeit mit verschiedenen Datenbanktabellen (mit SQL*Net kombiniert auch von verschiedenen Rechnern) sowie das Abspeichern von Prozeduren für den wiederholten Einsatz sind möglich.

Von Berichten über Briefe bis hin zu komplexen Dokumenten können mit dem SQL*Reportwriter (Bild 22) Ergebnisse mehrer Datenbankabfragen zusammengefaßt und ausgegeben werden. Die Bedienung erfolgt über Pull-down-Menüs; weitgehend selbsterklärende Eingabemasken und ein umfangreiches Hilfesystem unterstützen bei der Anwendung.

Für die professionelle (computergestützte) Softwareentwicklung (CASE) bietet Oracle das Paket **Oracle CASE**, unter anderem bestehend aus dem CASE*Dictionary, dem CASE*Designer und dem CASE*Generator.

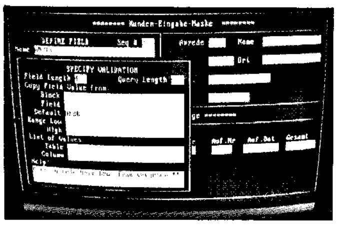


Bild 21

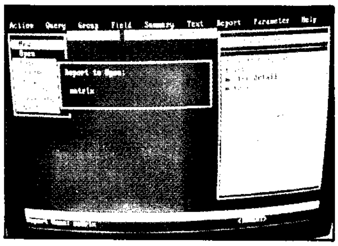


Bild 22

Ein oft gehörter Vorwurf ist der, das Softwareentwickler nur in der Lage seien, die Arbeit der anderen Computernutzer zu erleichtern, nicht aber ihre eigene, und so immer noch mit vorsintflutlichen Programmiertechniken arbeiten und damit keinesfalls die hohe Leistungsfähigkeit der von ihnen doch sonst so souverän beherrschten Technik auszunutzen verstehen. Dem soll durch CASE (Computer Aided Software Engineering) entgegengetreten werden. Oracle bietet mit seinen CASE-Tools Analyse- und Entwurfswerkzeuge, die den Entwickler bereits bei der Bedarfsanalyse unterstützen. Auch bei der Definition der Anforderungen an die konkrete Anwendungslösung, dem Design und der eigentliche Entwicklung bis hin zur Übergabe und zur Wartung wird dem Softwareentwickler Hilfe und Unterstützung gegeben.

Die Funktions- und Datenstrukturen, den Informationsfluß und die Bearbeitungsabläufe beim Entwurf und der Wartung von Datenbanken zu erfassen und zu verwalten, dient CASE*Dictionary. Mit diesem Hilfsmittel ausgerüstet soll der Entwickler frühzeitig auch die Auswirkungen möglicher Designänderungen analysieren können. Die Programmierung grafischer Oberflächen für die mit CASE*Dictionary entwickelten Anwendungen wird mit dem CASE*Designer möglich. Auch in Teamarbeit entwickelte, komplexe Lösungen können ansprechend gestaltet werden, was ja entscheidend zur Akzeptanz des Programms beiträgt. Die so entstandene Datenbankanwendung wird mit dem CASE*Generator in die entsprechenden Bildschirmmasken eingearbeitet, wodurch dem Entwickler zeitaufwendige und fehleranfällge Routinearbeiten abgenommen werden. Auch die mit Oracle CASE programmierten Anwendungen können über Standardnetze sowohl auf PCs als auch auf Workstations, Mini- und Großrechner verteilt werden.

Tafel 2

	SuperDB	SuperDB2
gleichzeitig verbundene Dateien	5	nur durch DOS begrenzt (FILES=)
Dateien je Verzeichnis	128	128
Datensätze je Datei	unbegrenzt	unbegrenzt
Felder je Datensatz	128	140
Zeichen je Feld	1028	1800
Zeichen je Satz	5048	2500
Klammerebenen	20	25
indizierbare Felder	99	120
Sortierebenen	30	40
Anzahl der Sortierkriterien	1000	unbegrenzt

TERMINE

XVII. Fachtagung des Fachausschusses 14

WER? Kammer der Technik, Fachverband Elektrotechnik
WANN? 22. November 1990
WO? Berlin, Kongreßhalle Alexanderplatz
WAS?
- speicherprogrammierbare Steuerungen bis zur Prozeßleittechnik
- Ausstellung von Hardware

WIE? Teilnahmemeldungen richten Sie bitte an:
Kammer der Technik, Präsidium, Fachverband Elektrotechnik, Clara-Zetkin-Straße 115/117, Berlin, 1086 *Hoppe*

2. Fachtagung Elektroakustik und Mikroelektronik '91

WER? Kammer der Technik, Fachverband Elektrotechnik
WANN? 6. und 7. März 1991
WO? Dresden
WAS?
- Auswirkungen elektrostatischer Entladungen auf die Elektronik
- theoretische Grundlagen
- Schutzmaßnahmen
- Prüfmethoden für den Nachweis der Wirksamkeit von Schutzmaßnahmen

WIE? Vortrags- und Teilnahmemeldungen richten Sie bitte an:
Kammer der Technik, Präsidium, Fachverband Elektrotechnik, Clara-Zetkin-Straße 115/117, Berlin, 1086 *Hoppe*

Bild 5. Postscript-fähiger Vierfarb-Thermotransfer-Drucker ColorScript 100 von QMS

Bild 6. Multimedia-Konzentrator CentreCOM 5000 von Allied Telesis Inc. und Terminalserver T151 von Spider Systems

Bild 7. Kassettenmagnetbandlaufwerke MA40 mit einer Kapazität von 1,3 GByte/Kassette (4 mm) von Micro Technology

Bild 8. Universelles Gefäßsystem der 6200-Serie für max. 18 Plattenspeicher (20 GByte) oder Kassettenmagnetbandlaufwerke MA24, MA40 oder MA92 von Micro Technology

Fortsetzung von der 2. Umschlagseite

68020-Mikroprozessor mit 4 MByte RAM. Die Auflösung beträgt 300 × 300 Pixel pro Zoll. Aus den vier Grundfarben gelb, rot, blau und schwarz werden auf speziellem Thermopapier 16 Mio. Farben zusammengemischt. Die Ausgabegeschwindigkeit beträgt ca. 4 A4-Seiten pro Minute.

Die SYNELEC Datensysteme GmbH, München, zeigte zahlreiche Kommunikationsprodukte von ISOLAN und SPIDER SYSTEMS wie z. B. den Terminalserver T151 (Bild 6) für das TCP/IP-Protokoll. Neu war insbesondere der Ethernet-Multimedia-Konzentrator CentreCOM 5000 (ebenfalls Bild 6) der Allied Telesis Inc. mit Repeatern für Thick-Wire-, Thin-Wire- und Glasfaserkabel.

Die Fa. Micro Technology GmbH, Wiesbaden, zeigte neben ihren „klassischen" Produkten wie Winchesterplattenspeicher und Ethernet-Kommunikationsprodukte vor allem das Kassettenmagnetbandlaufwerk MA92 mit einer Kapazität von 200 MByte/Kassette. Die Kassette ist kompatibel zur IBM-Kassette 3480 mit 18 Spuren. Das Aufzeichnungsformat unterscheidet sich aber bei IBM und DEC. Die Laufwerke können wahlweise mit einem Magazinwechsler für max. 10 Kassettenbänder geliefert werden. Weiterhin war neu im Angebot das Kassettenmagnetbandlaufwerk MA40 (Bild 7) mit einer Kapazität von 1,3 GByte/Kassette auf einer 4-mm-Kassette! Außerdem wurde ein neues Gefäßsystem der Serie 6200 (Bild 8) vorgestellt, das 18 Plattenspeicher (max. 20 GByte) sowie für Backupzwecke die Kassettenmagnetbandlaufwerke MA24, MA40 oder MA92 von Micro Technology aufnehmen kann.

Auf dem Gebiet der Massenspeichersubsysteme stellte sich als führender, unabhängiger Anbieter DEC-kompatibler Massenspeichersubsysteme die Fa. System Industries (Deutschland) GmbH vor. Ein Spitzenerzeugnis stellte das 3,5-Zoll-Winchesterplattenspeichersubsystem der SI350-Serie (Bild 9) mit einer Kapazität von 320 MByte/Laufwerk dar. Diese Laufwerke können als Kapazitätserweiterung für die VAXstation 3100 bzw. DECstation 3100 benutzt werden. Die mittlere Zugriffszeit beträgt 19,5 ms und die mittlere Zeit zwischen zwei Ausfällen (MTBF) ca. 150 000 Stunden (17 Jahre)!

Auf dem Gebiet der PC-Netze und ergänzender Hard- und Softwarekomponenten für Netze stellte sich als einer der führenden Anbieter die Fa. Compu-Shack Electronic GmbH, Neuwied, vor. Neben Anbieter von PCs der Fa. IBM, COMPAQ und PCs aus eigener Produktion ist Compu-Shack autorisierter Distributor für Novell-Netzwerk-Produkte in der Bundesrepublik. Darüber hinaus werden Ethernet-Interface-Leiterkarten für PCs (AT-Bus und Microchannel) aus eigener Produktion für Thick- und Thin-Wire-Ethernet (CS-Ethernet-Busmaster-Karten etc.) angeboten. Zur kompletten Ausstattung von PC-Fileservern gehört auch eine Unterbrechungsspannungsversorung für 10 bis 30 Minuten zum sicheren Abschluß aller Serveraktivitäten bei Netzspannungsausfällen bzw. zur Überbrückung kurzer Netzspannungsausfälle.

Prof. Dr. Thomas Horn

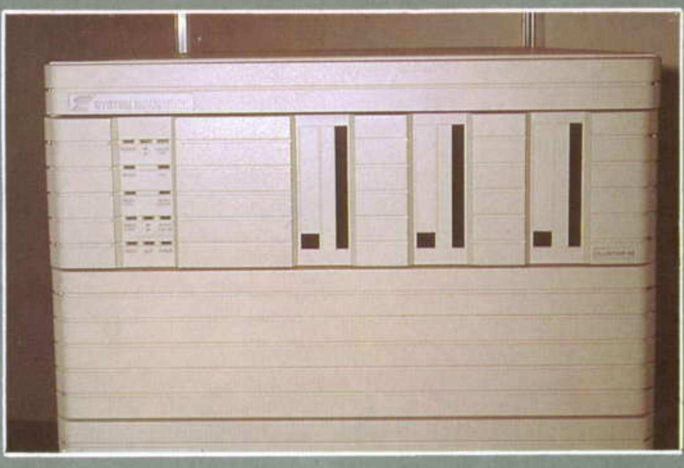


Bild 9. 3,5-Zoll-Plattenspeichersubsysteme SI350 mit 320 MByte/Laufwerk und Backup-Unterstützung von System Industries

ISSN 0232-2892
Mikroprozessortechnik, Berlin 4 (1990) 7
S. 193–224

vorgestellt

EC 1835 turbo

EC 1835-Transputer-Workstation

Hat Robotron bisher neue Büro- oder Personalcomputer im Abstand von mehreren Jahren vorgestellt, so waren zur diesjährigen Leipziger Frühjahrsmesse im März gleich drei neue Rechner der PC-Klasse zu sehen. Da wäre zuerst der zum IBM PC/AT-kompatible EC 1835 zu nennen, den wir bereits im vorigen Heft vorstellten. Für viele PC-Anwender sollte die Leistungsfähigkeit dieses 16-Bit-PCs, der international zur Zeit der Standard-PC ist, ausreichend sein. Aber auch für gehobenere Ansprüche – wie etwa CAD, Bildverarbeitung oder komplizierte und aufwendige mathematische Berechnungen – konnte Robotron Geräte dieser Rechnerklasse anbieten: seinen ersten 32-Bit-PC, den EC 1835 turbo, sowie die Transputer-Workstation auf der Basis des EC 1835, den EC 1835-TR.

Der EC 1835 turbo (Bild links) ist nicht eine schnelle Ausführung des 16-Bit-PCs EC 1835, wie man vermuten mag, sondern ein 32-Bit-PC mit dem Prozessor 80386 von Intel. Seine Taktfrequenz beträgt 25 MHz (und nicht wie angegeben 24 MHz). Der 1835 turbo kann mit einer 20-MByte- oder einer 40-MByte-Harddisk geliefert werden, er enthält ein 5,25-Zoll-Diskettenlaufwerk mit 1,2 MByte. Der EC 1835 turbo kann auf dem Motherboard mit bis zu 8 MByte RAM ausgestattet werden. Darüber hinaus besitzt er 7 programmierbare DMA-Kanäle und eine batteriegepufferte Uhr. Die zur Standardausrüstung gehörende Grafik reicht von einfacher Schwarzweiß-Grafik bis hin zu VGA (Video Graphics Array) mit 640 × 480 Bildpunkten und 16 Farben. Für Erweiterungen stehen 8 Steckplätze zur Verfügung. Als Betriebssysteme bietet Robotron sowohl das vom EC 1834 her bekannte, DOS-kompatible DCP 3.3 als auch das MS-DOS 4.01 an. Aber auch das neuste PC-Betriebssystem von Microsoft, das MS-OS/2 kann von Robotron bezogen werden. Hiermit wird die Arbeit im echten Multitasking möglich. Wer aber sogar Multiuserfähigkeiten des Rechners benötigt, der kann auf das Unix-kompatible Mutos 1835 zurückgreifen, das bereits auf dem EC 1835 lauffähig ist.

Die Transputer-Workstation EC 1835-TR8 (rechtes Bild) ist standardmäßig aus zwei Komponenten zusammengesetzt: aus dem EC 1835, der als Host-PC dient (im Bild rechts), und dem Transputer-Array EC 1835-TR. Der Host-PC ist ein EC 1835 mit einer Festplattenkapazität von mindestens 30 MByte, mit VGA-Grafik und mit der Transputer-Brückenkopfkarte EC1835-BRK. Diese enthält einen mit 20 MHz getakteten Transputer T800 und 4 MByte RAM. Dieser T800, der mit einer Link mit dem PC und mit drei Links mit dem Transputerarray verbunden ist, leistet 4 VAX-MIPS oder 10 Transputer-MIPS (Millionen Befehle pro Sekunde). Das Transputerarray befindet sich in einem EC 1834-Gehäuse (im Bild links) mit 8 Steckplätzen für die Knotenprozessorkarten EC1835-KNP mit je zwei T800 oder T414 von Inmos. Je nach der Zahl der bestückten Steckplätze ergibt sich die Gerätebezeichnung EC 1835-TR2 ... EC 1835-TR16. Die erreichbare Leistung liegt damit im Bereich von 8 bis 64 VAX-MIPS. Für alle Bestückungsvarianten ist die Netzwerktopologie frei wählbar.

Außer dem Host-PC können an das Netzwerk weitere EC 1835-TRxx-Netzwerke oder PC/AT angeschlossen werden.

Mit der vorgestellten Konfiguration EC 1835-TR8 soll eine Beschleunigung der Rechnerarbeit bis zum Faktor 50 möglich sein. Damit wird die Berechnung von umfangreichen und rechenintensiven Problemen mit einem vertretbaren Zeitaufwand möglich. Zur Demonstration der Einsatzmöglichkeiten dieser Transputer-Workstation stellte die Akademie der Wissenschaften der DDR das Programm FIDEFA V 1.0 zur Berechnung von Spannungs- und Deformationsfeldern in der nichtlinearen Statik vor. Es führt Berechnungen mit der Methode der finiten Elemente durch. Das Programm kann zum Lösen von Festigkeitsaufgaben verschiedener Art, aber auch zum Berechnen von Umformvorgängen eingesetzt werden. Die Materialien können metallisch, biologischer Art oder Plastmaterialien sein.

MP-Hk

Einige technische Daten

EC 1835 turbo	
CPU	i80386
Taktfrequenz	25MHz
Festplatte	20/40 MByte
Floppylaufwerk	5,25 Zoll, 1,2, MByte
Grafik	s/w oder VGA
Bildschirm	K 7228 für s/w, K 7233 für Farbe
Hauptspeicher	bis 8 MByte on board
Steckplätze	3 × 8 Bit, 4 × 16 Bit, 1 × 32 Bit
DMA-Kanäle	7, programmierbar
EC 1835-TR	
• EC 1835	
Festplatte	≥ 30 MByte
Grafik	VGA
Transputerkarte	EC1835-BRK
• EC1835-BRK	
Transputer	T800
Taktfrequenz	20 MHz
RAM	4 MByte
DMA-fähig, B004-kompatibel	
• EC 1835-TR2 ... EC 1835-TR16	
Gehäuse	EC 1834
Steckplätze	8
Transputerkarten	EC1835-KNP mit je 2 T800 oder T414
Knotenprozessoren	2 ... 16
RAM	8 ... 64 MByte
VAX-MIPS	8 ... 64
Topologie	frei programmierbar

ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

Heft 8 · 1990

Mikroprozessortechnik

Verlag Technik Berlin

ISSN 0232-2892

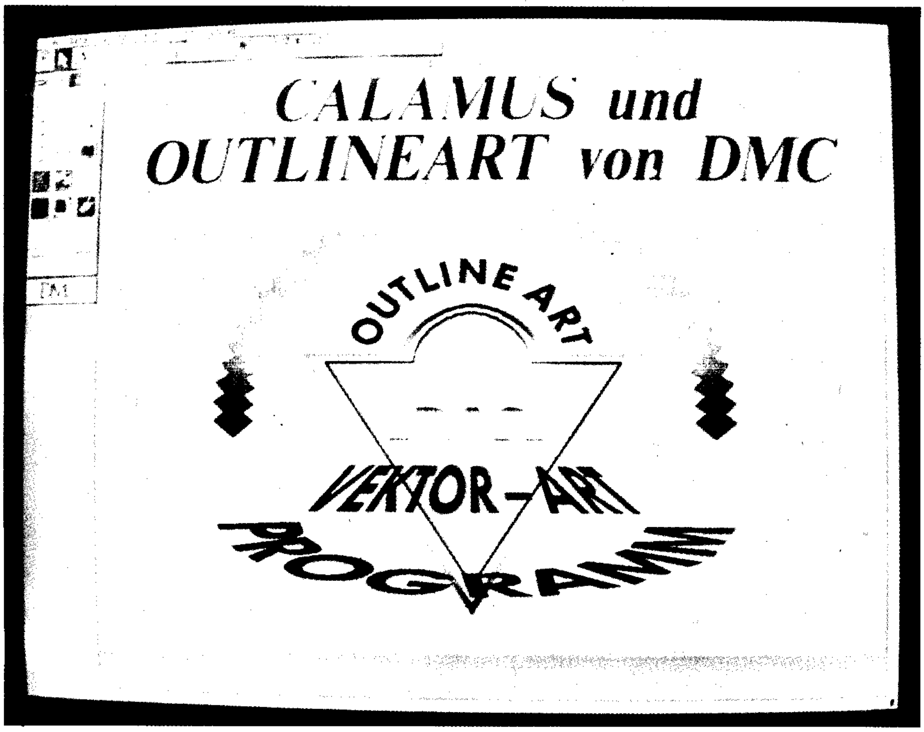
DM 5,-

mit Beiträgen aus c't

Spezialisten für die Signalverarbeitung:

Digitale Signalprozessoren

Calamus



Auf der Leipziger Frühjahrsmesse war am Stand von Atari auch das Atari-Dektop-Publishing-System **Calamus** von DMC zu finden. DMC steht dabei für Design Marketing Communication.

Da dieses Produkt zu den preisgünstigeren Angeboten im Bereich der DTP-Software gehört, möchten wir es Ihnen etwas näher vorstellen. Einziges Handicap: die Hardware – Voraussetzung ist ein Atari-Rechner. Und nicht nur irgendein Atari, etwa der PC 3, der PC 4 oder der ABC, sondern einer der Klasse Mega ST mit einem Motorola-Prozessor 68000 und möglichst mehr als 2 MByte Hauptspeicher. Darüber hinaus sollten ein hochauflösender, augenschonender Bildschirm Atari SM 124, eine 30- (besser: 60-) MByte-Atari-Festplatte und ein Laserdrucker Atari SLM 804 eingesetzt werden, sagt Atari. Von der Visitenkarte über Briefpapier und Werbeprospekte bis hin zu umfangreichen Handbüchern und sogar Katalogen wäre dann alles möglich.

Calamus soll durch ein anwenderfreundliches Konzept schnell beherrschbar und leicht zu bedienen sein. Ob allerdings den Anwendern heute der Umgang mit Calamus/DTP leichter fällt als seinerseits den Menschen im Altertum die Benutzung des gleichnamigen Schreibrohrs, kann aus naheliegenden Gründen leider nicht beantwortet werden. Ganz ohne Übung geht es aber dann sicher doch nicht, und so werden auch spezielle Schulungen und Serviceleistungen angeboten.

Die Texte können Sie über die Tastatur eingeben oder aber aus bereits bestehenden Dateien (auch aus solchen anderer Textverarbeitungsprogramme wie WordPerfect oder 1stWord) übernehmen. Der Text erscheint dann in jeder Phase der Bearbeitung nach dem sogenannten WYSIWYG-Prinzip so auf dem Bildschirm, wie Sie es dann auch nach dem Ausdruck auf dem Papier oder auch Film usw. erhalten werden. Es lassen sich somit nicht nur die Zeilen- und Absatzabstände frei wählen – das geht in bestimmtem Maße bereits mit einem *normalen* Textverarbeitungssystem –, sondern Sie sehen das auch schon auf dem Bildschirm anstelle der sonst dort üblichen Steuerzeichenfolgen. Den fertigen Text einschließlich der Tabellen, Grafiken und Fotos können Sie dann mit den verschiedensten Layout-Funktionen gestalten, was die immer noch weit verbreitete Schneide-Schiebe-Klebe-Technik in gewissem Maße ablösen kann. Eine Vielzahl möglicher Schriften steht dabei ebenfalls zur Auswahl. Die im sogenannten Vektorformat gespeicherten Zeichen können (auch einzeln) beliebig vergrößert, verkleinert und auch in verschiedenen Stilarten wie schattiert oder outlined (nur die Außenlinien) dargestellt werden. Feste Leerräume wie ein *Geviert* lassen sich dagegen nicht einstellen. Mehrere Zusatzpakete erweitern die Leistungsfähigkeit von Calamus, so beispielsweise *Outline-Art*, was unter anderem das Stauchen

STAUCHEN

den Farbverlauf eines Fonds

VERLAUF

den Satz entlang beliebiger Vektorpfade

CINEMASCOPE

aber auch das Verschieben, Rotieren, Spiegeln und die sphärische Projektion

KUGEL
KUGEL
KUGEL
KUGEL

von Texten, Grafiken oder Fotos ermöglicht. Mit dem *DMC-Lino-Interface* können Sie mit Calamus direkt – also ohne den sonst erforderlichen Raster-Image-Prozessor (RIP), der ein Postscript-File erzeugt und es pixelweise an den Laserdrucker schickt, Druckvorlagen herstellen und dabei die Möglichkeiten des Linotype-Laserbelichters nutzen. Das DMC-Lino-Interface kann neben dem Schriftpaket von Linotype auch mit denen von Agfa Compugraphic und URW geliefert werden. Das Zusatzpaket *Fonteditor* ermöglicht es, für Calamus eigene Zeichensätze mit mehr als 160 möglichen Zeichen, Objekten oder Piktogrammen aufzubauen, zu ändern, und ähnlich wie die Standardschiften auch in verschiedenen Stilarten in Calamus darzustellen.

Da das Calamus-Textverarbeitungssystem vermutlich nicht Ihren Ansprüchen genügt, bietet DMC als Ergänzung zum DTP-System den Texteditor *PKS-Write*, der in Zukunft in Calamus integriert werden soll. In sieben Dateien kann damit gleichzeitig gearbeitet werden, und es ist unter anderem möglich, Blöcke auszuschneiden, zu formatieren und zu sortieren, Texte und auch Schriftarten zu suchen und zu ersetzen sowie Makros ablaufen zu lassen. Sollten Sie Calamus allerdings professionell – beispielsweise in Druckbetrieben oder Belichtungsstudios – einsetzen wollen, dann bietet sich die *Calamus-Job-Version* an, die selbständig Joblisten für 99 Kunden pro Jobliste und 99 Dokumente pro Kunde abarbeitet, wobei die Druckoptionen für jedes Dokument speziell eingestellt werden können.

Alles in allem bietet DMC ein Softwarepaket, das trotz einiger Mängel zusammen mit der entsprechenden Hardware für 10 000 bis 15 000 DM professionelles Desktop-Publishing ermöglicht und damit für etwa die Hälfte bis zu einem Drittel des Preises anderer Systeme (Macintosh, IBM PC/AT) Vergleichbares leistet.

MP-HI

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
4. Jahrgang · ISSN 0232–2892

Mikroprozessortechnik, Heft 8 · 90

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag Verlag Technik GmbH, Oranienburger Str. 13/14, DDR–1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd, Telefax: 2 87 02 59

Geschäftsführer Klaus Hieronimus

Redaktion Hans Weiß, Chefredakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2 87 02 03), Hans-Joachim Hill (Tel. 2870209); Sekretariat: Hannelore Kulik (Tel. 2870381)

Gestaltung Christina Bauer

Titelfoto Thomas Bölkow

Beirat Prof. Dr. sc. Thomas Horn, Prof. Dr. Bernd Junghans, Dr. Dietmar Keller, Prof. Dr. sc. Bernd-Georg Münzer, Prof. Dr. sc. Peter Neubert, Prof. Dr. sc. Dr. Michael Roth (Vorsitzender), Dr. Gerhard Schulze, Prof. Dr. sc. Manfred Seifart, Dr. Dieter Simon, Prof. Dr. sc. Dr. Jürgen Zaremba

Lizenz-Nr. 1710 des Presse- und Informationsdienstes der Regierung der DDR

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

Anzeigenannahme Verlag Technik GmbH, Anzeigenabteilung, Oranienburger Straße 13/14, Berlin, 1020; Tel. 2 87 02 54, 2 87 03 09, Telefax: 2 87 03 09

Anzeigenpreise Preisliste Nr. 1 vom 1. Juli 1990

Erfüllungsort und Gerichtsstand Berlin-Mitte. Der Verlag behält sich alle Rechte an den von ihm veröffentlichten Aufsätzen und Abbildungen, auch das der Übersetzung in fremde Sprachen, vor. Auszüge, Referate und Besprechungen sind nur mit voller Quellenangabe zulässig.

Redaktionsschluß 14. Juni 1990

AN (EDV) 49837

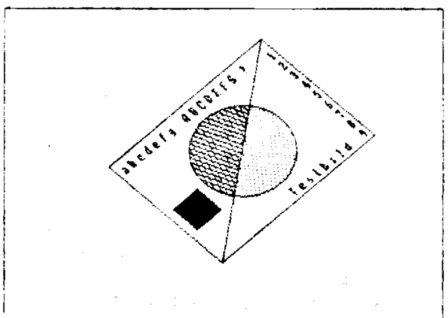
Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– DM, Abonnementspreis vierteljährlich 15,– DM; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

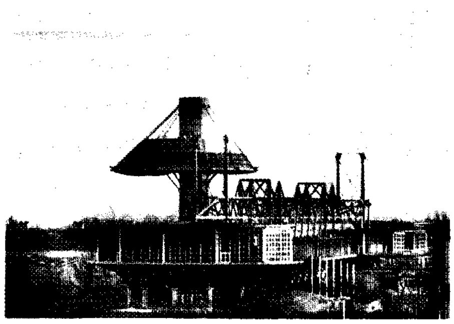
Bezugsmöglichkeiten
Verlag Technik, Abteilung Absatz, PSF 201, Berlin, 1020 oder: *DDR:* sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSFR:* PNS – Ustřední Expedicia a Dovoz Tisku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *Republik Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Republik Ungarn:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nội; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Buchhandel oder Verlag Technik GmbH

Zum Titelbild

Mikroprozessoren benötigen für die Lösung von Echtzeitaufgaben – wie digitale Filterung in der Sprachverarbeitung oder bei der Datenübertragung – hohe Rechengeschwindigkeiten, besonders bei der Multiplikation und der Datenbereitstellung für die Verarbeitung in der arithmetischen Einheit. Diese Aufgaben können „spezialisierte" Mikroprozessoren übernehmen. Der Beitrag „Digitale Signalprozessoren" auf der **Seite 4** stellt diese Spezialisten vor. Er geht auf die innere Struktur der DSPs ein und vergleicht ihre verbreitetsten Vertreter.



Es kann vorkommen, daß gespeicherte Bilder neueren Anforderungen nicht mehr gerecht werden. Notwendig könnten andere Formate oder Drehungen werden. Der Beitrag „Rasterbilder auf dem PC" auf der **Seite 13** stellt deshalb Algorithmen vor, die die weitere Bearbeitung von Bildern gestatten.



Ganz so futuristisch, wie sich das neue Tagungs-Centrum Messe (TCM) dem CeBIT-Besucher präsentierte, ging es bei den Exponaten zwar nicht zu. Dennoch glauben wir, daß es genügend Neuvorstellungen gab, die trendbestimmend sein werden. Eine Auswahl davon finden Sie in unserem CeBIT-Bericht als Nachlese – ab **Seite 33**.

Inhalt

vorgestellt — **2. US**
Calamus

MP-Info — **2**

Gert Heuer:
Digitale Signalprozessoren — **4**

Elke Badermann, Dieter Kühnel, Siegfried Löbbicke, Peter Piontek:
Peripherieschaltkreise der Reihen DS 8600 und DL 8600 — **10**

Peter Döring:
Rasterbilder auf dem PC — **13**

MP-Kurs: — **19**
Wolfram Schulze, Uwe Schulze:
Einführung in MS-DOS (Teil 3)

kennengelernt — **26**
Knut Löschke:
Quick-Basic 4/4.5

Jürgen Pietsch:
Kopierbahnhof VGA-RAM — **28**

Wegbereiter der Informatik
Howard Hathaway Aiken — **29**

Turbo-Pascal-Tip:
Heller Hintergrund im Textmodus — **29**

MP-Computer-Club — **30**
Christian Löber:
BICLAN – Ein lokales Netz für Lehrkabinette

vorgestellt — **31**
Commodore Amiga 3000

MP-Literatur — **32**

MP-Bericht — **33**
CeBIT '90

vorgestellt — **4. US**
Seikosha SL-532

Vorschau

Für Heft 9/1990 bereiten wir für Sie unter anderem Beiträge zu folgenden Themen vor:

- MS-DOS und Multitasking
- Computerviren
- Unix-Prozeßkommunikation
- Hardware-Interrupts beim A 7150

VDI-Richtlinien in der DDR erhältlich

Mit Gründung der Vereinigung „Verein Deutscher Ingenieure – Gliederung DDR“ am 11. März 1990 in Leipzig sind jetzt die VDI-Richtlinien in der DDR erhältlich. Die VDI-Richtlinien sind ein wesentlicher Bestandteil des deutschen technischen Regelwerks und konkrete Arbeitshilfen für den im Beruf stehenden Ingenieur. Sie stellen technische Sachverhalte praxisbezogen dar und dokumentieren den Stand von Wissenschaft und Technik.
Die VDI-Richtlinien sind in Handbüchern zusammengefaßt, von denen unter anderem folgende vorliegen: Betriebstechnik, Konstruktion, Meßtechnik I und II, Regelungstechnik, Technische Zuverlässigkeit, Bürokommunikation, Werkstofftechnik.
Bestellungen können an den Verlag für Standardisierung, Wallstr. 20, 1020 Berlin, gerichtet werden.

Weiterbildungsinstitut

Das Institut für Datenverarbeitung und Betriebswirtschaft (IDB), Frankfurt (Main), hat am 15. Mai 1990 ein eigenes Büro und Schulungsräume im Technologie- und Ausstellungszentrum (TAZ) in Berlin-Wuhlheide eröffnet. Die bereits in den Vormonaten erfolgreich durchgeführten Weiterbildungsveranstaltungen werden nun fortgesetzt.
Das Schulungsangebot des IDB umfaßt alle Bereiche der Betriebswirtschaft und der Datenverarbeitung. Weitergehende Informationen sind direkt im Büro des IDB im Technologie- und Ausstellungszentrum, Köpenicker Straße 325, Berlin, 1170, Tel. 65 76-29 75 zu erhalten.

Kontron Elektronik in Leipzig

Die Kontron Elektronik GmbH in Eching bei München hat Anfang Mai ein Technisches Büro in Leipzig eröffnet, in dem Mitarbeiter aus der DDR, die in Eching geschult wurden, Vertrieb und Service von Kontron-Produkten übernehmen. Kontron Elektronik stärkt damit seinen Vertrieb in einem Markt, in dem das Unternehmen seit März 1990 leistungsfähige Industrierechner und hochwertige CAD/CAM-Systeme an Industrieunternehmen und Forschungsinstitute verkauft. Die Firma entwickelt, produziert und vertreibt Produkte in den Bereichen Meßtechnik, Datensysteme, Bildanalyse/Bildverarbeitung und Optoelektronische Sensorik.
Kontaktadresse: Kontron Elektronik, TB Leipzig, Breite Straße 25, Postfach 9, 7245 Naundorf b. Leipzig; Tel. 00 33 41/22 82

Kostenlose Datenbank

In einer gemeinsamen Aktion wird von den Unternehmen ODIN IB Lohrmann (Karl-Marx-Stadt), SOS Software Service GmbH (Augsburg) und ComWare (Berlin) das programmierbare Datenbanksystem Index kostenlos abgegeben. Das Programm basiert auf dem Novell-Datenbanksystem Btrieve und wurde in Modula-2 geschrieben. Mit Index sollen auch DV-Laien komplexe Applikationen in einer einfachen Programmiersprache erstellen können. Die Haupteinsatzgebiete liegen im kommerziellen Bereich, beispielsweise in der Kunden- und Lagerverwaltung und Fakturierung. Im LAN-Einsatz werden die Lockingmechanismen automatisch unterstützt. Index läuft unter den Betriebssystemen MS-DOS und DCP.
Das Programm und das komplette Handbuch befinden sich auf zwei Disketten, die gegen Rückporto von ODIN IB Lohrmann, PSF 7, Chemnitz, 9061, zu beziehen sind.

Basic PDS 7.0

Allen Besitzern von Quick-Basic bietet Microsoft jetzt, und solange der Vorrat reicht, das professionelle Entwicklungssystem Basic PDS 7.0 zu dem Upgrade-Preis von 750,– DM (Listenpreis: 1357,– DM) an. Gegen Einsendung des alten Produkts und eines Upgrade-Coupons bzw. der Registrierkarte und einer Kopie der Rechnung. Damit soll den zahlreichen professionellen Basic-Programmierern eine Leistungsfähigkeit ähnlich der anderer Hochsprachen wie C und Pascal erschlossen werden.
Microsoft GmbH, Edisonstraße 1, D-8044 Unterschleißheim; Tel. (089) 3 17 05-0 *MP*

EXPO COMM MOSKAU '91

Erste internationale Fachmesse der Informations- und Telekommunikationsindustrie in der Sowjetunion

Zum erstenmal wird es 1991 in der Sowjetunion eine internationale Fachmesse geben, die das gesamte Spektrum der Informations- und Telekommunikationsindustrie abdeckt. Hannover-Messe International, eine Tochtergesellschaft der Deutschen Messe AG. Hannover, und die internationalen Messeveranstalter E. J. Krause & Associates, Washington, D. C. richten vom 5. bis 10. Juni 1991 die EXPO COMM '91 auf dem All-Unionsausstellungsgelände in Moskau aus.
Angesichts der zu erwartenden Lokkerung der COCOM-Bestimmungen und des Umgestaltungsprozesses in der Sowjetunion ist auch mit neuen Akzenten in der Investitionspolitik zu rechnen. Schon heute ist das Ziel, die soziale Infrastruktur z. B. durch moderne Kommunikationstechnik zu verbessern, im Volkswirtschaftsplan festgeschrieben. Unter anderem soll der Umfang der Leistungen des Post- und Fernmeldewesens um mindestens 26 Prozent erhöht werden. Für die exportorientierte Informations- und Telekommunikationsindustrie sind dies günstige Voraussetzungen für einen erfolgreichen Messeverlauf. Die Anmeldeunterlagen und weitere Informationen sind erhältlich bei: *HANNOVER-MESSE INTERNATIONAL GmbH, Unternehmensgruppe Deutsche Messe AG, Messegelände, D-3000 Hannover 82, Tel. 89-3 14 00 und 89-3 14-07.* *MP*

Halbleiterbauelemente-Informationen

Insbesondere bislang selten in der DDR eingesetzte internationale Bauelemente sind hierzulande noch wenig spezialisiert. Die Firma MCT hat daher für alle interessierten Elektronikanwender einen Datenblattservice eingerichtet. Gegen eine Gebühr kann man sämtliche vorhandene Unterlagen vor Ort kopieren. Interessenten steht die firmeneigene Bibliothek in der Leipziger Simsonstraße 9 dienstags, mittwochs, und donnerstags von 16.00 bis 19.00 Uhr zur Verfügung.
MCT Leipzig vertreibt Produkte der Partnerfirmen MCT Berlin und Elektronikladen Detmold. Neben leistungsfähigen universellen Einplatinenrechnern handelt es sich dabei um PC- und VME-Bus-Baugruppen, für welche auch Software entwickelt und angeboten wird. *MP*

TommySoftware legalisiert Raubkopien

Das Westberliner Unternehmen TommySoftware legalisiert mit Wirkung vom 1. Juli 1990 sämtliche Raubkopien seiner Softwareprodukte in der DDR. Voraussetzung ist der kostenlose Erwerb einer Registrationsnummer bei dem Unternehmen im Schriftverkehr bis 30. September 1990. Alle später eingehenden Registrationen können nicht mehr berücksichtigt werden.
Kontakt: TommySoftware, Selchower Str. 32, D-1000 Berlin 44; Tel. 62 14 06-3

Beschränkungen im Osthandel gelockert

Seit Mai sind die Exportbeschränkungen für die Lieferung von Seagate-Festplatten und -Controllern in den Ostblock teilweise aufgehoben. Wie Seagate bekanntgab, bestätigte das für die Außenhandelskontrolle zuständige „United States Department of Commerce“, daß die Ausfuhr bestimmter 3,5- und 5,25-Zoll-Laufwerke nach Osteuropa keinen Beschränkungen mehr unterliegt. Alle Modelle mit einer Kapazität unter 55 MByte – unformatiert – und einer Datentransferrate unter 5,2 MBit/s können jetzt ohne Einschränkungen geliefert werden. Hierzu gehören unter anderem die Laufwerke ST 225, ST 251, ST 151 und ST 125.
Nach Ansicht von Seagate ist dies der erste Fall, bei dem ein Antragsteller aus der Industrie die Freigabe von Harddisks nach den amerikanischen „Foreign-Availability“-Vorschriften erfolgreich durchsetzen konnte. *MP*

Zusammenarbeit der BRD- und DDR-Elektronikindustrie

Viel entschlossener Wille zum aktiven industriellen Engagement in der DDR, aber auch unverkennbare Zurückhaltung der bundesdeutschen Unternehmen angesichts zahlreicher noch bestehender Unsicherheiten im politischen und rechtlichen Raum: In dieser Grundstimmung fand im Mai die diesjährige Jahres-Mitgliederversammlung des Fachverbandes Bauelemente der Elektronik im Zentralverband Elektrotechnik- und Elektronikindustrie (ZVEI) statt. An dieser Begegnung der deutsch-deutschen Bauelementeindustrie nahmen über hundert Bauelemente-Hersteller aus der Bundesrepublik und Repräsentanten der Elektronikindustrie der DDR teil. Unter dem Motto: „Ein Staat ändert seine Wirtschaftsform – Die Chancen der Bauelementeindustrie für Kooperation und Handel mit der DDR“ diskutierten die Firmenvertreter aus Ost und West über Notwendigkeit und konkrete Möglichkeiten der industriellen Zusammenarbeit zwischen den Unternehmen in beiden Teilen Deutschlands. Breiten

MP-Dialog

Schnell, schneller, Turbo-Pascal?

Zu unserem Artikel **„Neuer Funktionsinterpreter in Turbo-Pascal“** (*MP 3/90, Seite 72*) erhielten wir von Herrn Rainer Schnauß aus Nordhausen den Hinweis, daß man die Rechenzeit des vorgestellten Programmes für lange Formeln auf etwa die Hälfte reduzieren kann, wenn man die Priorität der Operationen analysiert und die Formel umkodiert. „Dazu wird jeder implementierten Funktion ein Zeichen zugeordnet, und die in der Formel vorkommenden Funktionen werden durch ihre zugehörigen Zeichen ersetzt. Jede ermittelte Zahl wird in einer linearen Liste abgelegt und ihre Stelle in der Formel durch ein definiertes Zeichen markiert. Während dieser Umkodierung werden bereits Plausibilitätsprüfungen durchgeführt.
In einem zweiten zeichenweisen Durchlauf durch die umkodierte Formel werden die restlichen Prüfungen auf syntaktische Richtigkeit durchgeführt, die während der Umkodierung noch nicht realisiert wurden. Auf diese Art werden insbesondere auch alle syntaktisch falschen Formeln, die der im Heft 3 vorgestellte Funktionsinterpreter noch durchgehen läßt, verworfen.“
Die erreichte Einsparung muß allerdings mit einer erheblichen Verlängerung des Quelltextes erkauft werden.
Herr Jörg Minkwitz aus Berlin möchte den Beitrag **„Programmiersprachen für MS-DOS“** (*MP 4/90, Seite 103*) mit dem Hinweis ergänzen, daß sich die Laufzeiten von Turbo-Pascal-Programmen durch das Setzen der Compiler-Schalter { $ R-, S- } wesentlich verkürzen lassen. Ein erneuter Test mit gesetzten Schaltern ergab für die Fibonaccifunktion eine Verkürzung von 1:13 min auf 0:49 min. *MP*

Raum nahm auch die Diskussion über die Einordnung einer künftigen gesamtdeutschen Bauelementeindustrie im europäischen und internationalen Kontext ein. Jürgen Knorr, Vorsitzender des Fachverbandes Bauelemente der Elektronik, zur traditionellen Vorreiterrolle der Industrie gegenüber der Politik: „Der ZVEI-Fachverband Bauelemente der Elektronik und seine Mitgliedsfirmen sind bereit, mit einem aktiven Engagement in der DDR nicht bis zur staatlichen Einheit Deutschlands zu warten. Die DDR spielt in den Überlegungen westdeutscher Firmen nicht nur als Vertriebsmarkt, sondern auch als Produktions- und Entwicklungsstandort eine wichtige Rolle."
Im Fachverband Bauelemente der Elektronik des ZVEI sind rund 140 Hersteller elektronischer Bauelemente in der Bundesrepublik Deutschland zusammengeschlossen.

2. ASIC-Seminar

Am 2. und 3. Mai 1990 fand an der TU Karl-Marx-Stadt das 2. *ASIC-Seminar* statt, ein Problemseminar zum Entwurf anwendungsspezifischer Schaltkreise. Dazu hatten der WB Mikroelektronikentwurf der TU, der Fachausschuß ASIC der KDT und das Schaltkreiszentrum des VEB Textimaelektronik eingeladen.
Das Seminar stand im Zeichen sich ändernder gesellschaftlicher und wirtschaftlicher Rahmenbedingungen in der Elektronik-Industrie der DDR. Das drückte sich zum einen in der im Vergleich zum 1. ASIC-Seminar 1988 etwas geringeren Teilnehmerzahl und zum anderen im Vortragsangebot und in Diskussionen zu Entwurfs- und Anwendungsmöglichkeiten der ASICs aus. In 15 Vorträgen vermittelten die Referenten ein breit gefächertes Bild, von der Vorstellung neuer ASIC-Master über die Anwendung neuer und bewährter Entwurfshilfsmittel für Entwurf und Einsatz von ASICs.
Im Plenarvortrag stellte *Prof. Müller* (TUK) die verschiedenen Arten der ASICs dar, erläuterte die wachsende Notwendigkeit des Einsatzes funktioneller Beschreibungsmöglichkeiten, um die steigende Komplexität der ASICs zu beherrschen und verwies auf internationale Tendenzen in der ASIC-Entwicklung.
Dr. Wolf (TEXEL), *Dr. Schmidt* (CZJ) und *Seiler* (CZJ) berichteten über Vorstellungen zur ASIC-Entwicklung in der DDR aus firmenspezifischer Sicht und über den Aufbau von Softmacrobibliotheken. Diese Ausführungen wurden von *Dr. Ritter* (MEE) zu dem in Erfurt entwickelten Gate-Array-System U 6xxx ergänzt. *Pohland* (HFO) stellte ein in Frankfurt entwickeltes bipolares Gate-Array-System in ECL-Technik vor.
Breiten Raum nahmen Vorträge zur Softwareunterstützung im Gate-Array-Entwurf ein. *Dr. Troll* (TEXEL) berichtete über grafische Beschreibungsmittel im Schaltkreisentwurf, *Dr. Koegst* (AdW) über den Steuerwerksentwurf mit MIPRE und *Dr. Donath* (AdW) über Einsatzerfahrungen mit dem Simulationsprogramm KOSIM beim ASIC-Entwurf. Eine Gruppe von Vortragenden des WB Mikroelektronikentwurf der TUK stellte neue Entwurfshilfsmittel vor.
Dr. Steinbach gab einen Überblick über den Logik-Entwurf mit dem Logiksynthese-Modul ALOE, der von *Landmann* mit speziellen Ausführungen zur Anwendung von ALOE für den ASIC-Entwurf ergänzt wurde. Um insbesondere große komplexe digitale Gate-Array-Systeme entwerfen zu können, macht sich die Nutzung funktioneller Beschreibungen für den Entwurf notwendig. Dazu wurden von *Knoth* (TUK) Erfahrungen und Lösungen vorgetragen. Die Vorstellung des neuen Programmsystems PC-GAD-3 zum Entwurf von Gate-Array-Schaltkreisen auf der Basis des Masters U 5300 mit Hilfe der PC-Rechentechnik war Inhalt eines Übersichtsvortrages von *Pauliuk* (TUK).
Die Teilnehmer des Seminars waren sich nach einem abschließenden Gespräch einig, die Tradition der ASIC-Seminare fortzuführen. Neben der Vermittlung von Anwendererfahrungen und der Vorstellung neuer Entwurfshilfsmittel sollte auch der Darstellung international verbreiteter Gate-Array-Entwurfssysteme mehr Aufmerksamkeit geschenkt werden.
Zum nächsten ASIC-Seminar wird rechtzeitig durch die Veröffentlichung in der Fachpresse eingeladen.

Dr. Brumme

1. Berliner Unix-Forum

Die Jahrestagung der Entwickler- und Anwendergemeinschaft Unix-kompatibler Betriebssysteme (Unix-EAG) am 22. Mai in der Ostberliner Kongreßhalle stand in diesem Jahr ganz im Zeichen der sich abzeichnenden deutsch-deutschen Zusammenarbeit und wurde mit einer Ausstellung führender Hard- und Softwareanbieter aus der Bundesrepublik zum 1. Berliner Unix-Forum vereinigt. Für die Ausstellung, bei der längst nicht alle Interessenten berücksichtigt werden konnten, hatten folgende Unternehmen Stände zwischen 6 und 18 m^2 aufgebaut: AEG Olympia/EAW, Altos Computer Systems, Apple Computer, Arrow, Beech Tree Systems, BinTec Computersysteme, BSP Softwaredistribution, Bull, Concept ASA, Concurrent Computer, Digital Equipment, Garmhausen und Partner, gki/UniWare Computer GmbH, Heise-Verlag, Hewlett-Packard, IDG Verlag, MAI, Mannesmann Kienzle, Markt & Technik, Motorola, Mummert & Partner, NCR, Nomina GmbH, TA Olivetti, PSC Computer Systeme, PSI, Quadraton Deutschland, Sequent Computer, Siemens, Sony Microsystems, Stardent Computer, The Santa Cruz Operation, Unisys Deutschland. Daß ein klingender Name fehlte, nämlich IBM Deutschland, lag nicht am fehlenden Bemühen der Organisatoren, sondern wohl eher an den zu diesem Zeitpunkt noch geltenden COCOM-Restriktionen.
Nach Eröffnung der Jahrestagung und dem Grußwort der GUUG (German Unix User Group) wurde zunächst ein Überblick über die Hardwaresituation im Unix-Bereich gegeben. Danach beantwortete ein Vertreter des EAW die viele im Saal bewegende Frage, wie es mit der P8000-Technik weitergehen würde. Zwei weitere aus dem Kreis der EAG gehaltene Vorträge beschäftigten sich mit dem Aufbau eines Krankenhaus-Informationssystems sowie mit der Gestaltung von Multiprozeßsystemen. Der zweite Teil des Vortragsprogramms wurde in diesem Jahr Gastrednern jener Einrichtungen überlassen, die in der Standardisierungsdiskussion um Unix eine führende Rolle spielen. Dabei hatte AT&T Gelegenheit, über das Unix-System V, Release 4 zu referieren, X/Open über Brennpunkte der Entwicklung zu offenen Systemen sowie OSF über das Betriebssystem OSF/1. Nach der Jahrestagung blieb noch genügend Zeit, mit dem einen oder anderen Aussteller ins Gespräch zu kommen oder zu einem Gedankenaustausch unter Kollegen und Mitgliedern der EAG. Die Meinungen über die Veranstaltung waren bei den meisten der etwa 1500 Besucher sehr positiv. Eine Befragung der Aussteller ergab bereits mittags, daß ihre Erwartungen voll erfüllt, bei vielen sogar weit übertroffen waren. (Das wurde vor allem deutlich bei denen, die schon zeitig ohne Werbematerial dastanden!) Bei der gki# mbh, die als Dienstleistung die EAG betreut, und bei UniWare#, die die Ausstellung organisiert hatte, beginnen demzufolge bereits die Vorbereitungen für das 2. Berliner Unix-Forum 1991.

Schenck

13. DECUS-München-Symposium

Das 13. Symposium der Anwenderorganisation DECUS München e. V. für Anwender elektronischer Klein- und Mikrorechenanlagen der Digital Equipment Corporation (DEC), Maynard/USA, fand vom 12. bis 16. März 1990 in Würzburg statt. DECUS München e. V. ist mit über 8000 Anwendern aus dem deutschsprachigen Raum Bestandteil der weltweiten Digital Equipment Computer User Society (DECUS) mit weit über 100 000 Mitgliedern. Hauptziel des Symposiums ist neben der Wissensvermittlung über Hard- und Softwareprodukte für DEC-Rechner die Weiterentwicklung des Erfahrungsaustausches zwischen den Anwendern sowie die Rückkopplung der Einsatzerfahrungen zu den Entwicklern.
Am ersten und letzten Tag bestand die Möglichkeit der Teilnahme an eintägigen Trainingskursen zu den Problemkreisen VAX/VMS-Systemmanagement, VAX-Cluster-Grundlagen für Systemmanager, DECnet-Management, Sicherheit in verteilten Systemen u. v. a. Themen. An den eigentlichen Symposiumstagen wurden in teilweise bis zu 10 parallelen Sitzungen umfangreiche Informationen zu Hard- und Software sowie Anwendungserfahrungen vermittelt. In Workshops konnten Probleme und Einsatzerfahrungen diskutiert werden. In den Q&A-Sitzungen (Question and Answer) konnten Anfragen an die DEC-Mitarbeiter, insbesondere aus dem Engineering und Telefon Support Center gerichtet und Probleme zur Beantwortung weitergeleitet werden.
Die über 190 Vorträge wurden entsprechend der Struktur von DECUS in den SIGs (Spez. Interessengruppen) gehalten wie SIG Große VAX-Systeme, SIG Netzwerke, SIG Betriebssystem VAX/VMS, SIG Betriebssystem UNIX, SIG Betriebssystem RSX-11, SIG Betriebssystem RT-11, SIG Image- und Grafik-Processing, SIG CAD/CAM-Anwendungen, SIG Sprachen und Werkzeuge, SIG Büroautomatisierung u. a.
Besonderes Interesse der Teilnehmer fanden die Vorträge zur Vorstellung der neuen VAX-Systeme wie VAX 6000, Modelle 410...460 und VAX 9000, Modelle 210...440 sowie die Vorträge über die Phase V von DECnet und die Vorstellung neuer Netzwerkprodukte.
Auf dem Unix-Sektor wurde die neue Ultrix-Version 4.0 vorgestellt. In der SIG Sprachen und Werkzeuge waren vor allem die Vorträge zu OSF (Open Software Foundation) und Weiterentwicklung des Betriebssystems VAX/VMS zur Version 6 auf der Basis der Posix-Standards von Interesse. Darüber hinaus gab es auch auf eine Vielzahl anderer Sitzungen wertvolle Informationen und Hinweise zur Weiterentwicklung von anderen Softwareprodukten.

Prof. Dr. Horn

Spezialist für Bürokommunikation in der DDR

Um DDR-Bürgern den Zugang zu Kopierern, Telefaxgeräten, Computern und elektronischen Schreibmaschinen zu erleichtern, plant die UTAX GmbH, in mehreren Städten der DDR Dienstleistungszentren aufzubauen. Dort sollen zunächst einmal UTAX-Kopier- und Telefaxgeräte, PCs und Schreibsysteme bereitgestellt werden, so daß sie von jedermann genutzt werden können. Längerfristig soll dies, so die Firmenleitung, die Basis für die Entwicklung eines UTAX-Vertriebsnetzes auch in der DDR sein.
Für den Anfang ist das Unternehmen auf der Suche nach Räumlichkeiten, in denen die ersten UTAX-Zentren entstehen können. Außerdem sucht UTAX Personal, das den Nutzern des Services mit Rat und Tat zur Seite steht und sich um Organisatorisches wie die Wartung der Geräte kümmert. Bislang nicht vorhandenes Wissen würde durch entsprechende Schulungsprogramme vermittelt werden.

Die Kontaktadresse lautet: UTAX GmbH, Erwin Lehmann, Ohechaussee 235, D-2000 Norderstedt; Tel. 52 84 90, Fax: 0 40/52 84 01 20, Telex: 04 05 28 49 utaxd.

MP

Verkaufspartner von Siemens-Tochter in der DDR

Als autorisierter Verkaufspartner der Siemens-Tochter Computer Gesellschaft Konstanz mbH in der DDR hat sich der ComputerService Magdeburg (CSM) – ehemals Datenverarbeitungszentrum – seinen langjährigen sowie potentiellen Kunden vorgestellt. Der Magdeburger Betrieb wird künftig eigenständig Erzeugnisse der Siemens-Tochter vermarkten und für sie den gesamten Kundendienst auf dem Gebiet der DDR übernehmen. Es handelt sich zunächst um optische Belegleser, für die anwenderorientierte Systemlösungen angeboten werden.
Für die Zukunft denke man auch daran, einzelne Systemkomponenten in der DDR produzieren zu lassen, wozu der Magdeburger Betrieb die Lizenzvergabe organisieren werde.

ADN

Digitale Signalprozessoren

Gert Heuer, Berlin

Die Entwicklung universeller Mikroprozessoren ermöglichte Anfang der siebziger Jahre eine wesentliche Steigerung der Leistungsfähigkeit digitaler Schaltungen und war somit eine wesentliche Voraussetzung zur Dezentralisierung und Miniaturisierung der digitalen Rechentechnik. Die Struktur der Mikroprozessoren unterstützt eine universelle Softwareentwicklung durch vielfältige Adressierungsmöglichkeiten, Speicher- und Registeroperationen unter Verwendung eines großen Kellerspeichers (Stack) und einer Vielzahl von Registern. Für die effektive Umsetzung von Signalverarbeitungsalgorithmen besitzen die Mikroprozessoren jedoch eine viel zu geringe arithmetische Leistungsfähigkeit. Dies betrifft vor allem die Rechenzeiten für die Multiplikation und die Datenbereitstellung für die Verarbeitung in der arithmetischen Einheit.
Für den Einsatz von Prozessoren in der Signalverarbeitung wurde deshalb die Entwicklung spezieller, digitaler Signalprozessoren erforderlich, die wesentlich kostengünstigere und leistungsfähigere Konzepte ermöglichen. Die Eigenschaften digitaler Signalprozessoren, wie die schnelle Befehlsausführung von arithmetischen Operationen, zum Beispiel die Multiplikation in nur einem Befehlszyklus, die parallele Adreßrechnung über Adreßrechenwerke, die Multiprozessorfähigkeit und das Vorhandensein einer seriellen Schnittstelle, eröffnen einen breiten Einsatz dieser Prozessoren in der Signalverarbeitung. Die sich ständig verbessernden Leistungsparameter der digitalen Signalprozessoren führte in den letzten Jahren zur Erschließung immer neuer Anwendungsgebiete der Digitaltechnik in der Industrie und in der Konsumgüterelektronik sowie zur Entwicklung völlig neuer Erzeugnislisten. So soll das Verkaufsvolumen der DSPs nach einer Marktstudie der Frost & Sullivan Incorporation in den USA bis 1991 jährlich um 30 % steigen.
Einsatzgebiete der digitalen Signalprozessoren sind unter anderem:
– Automatisierungstechnik (Steuer- und Regelungstechnik)
– Nachrichtentechnik (Modem, Telecom)
– Informationstechnik (Bildverarbeitung, Spracherkennung)
– Meßtechnik (Spektralanalyse, Signaltransformation).
Diese Einsatzgebiete sind oft dadurch gekennzeichnet, daß sie besonders hohe Anforderungen an den Datendurchsatz und die Echtzeitfähigkeit des Systems stellen. Deshalb ist für die Entwicklung und die Anwendung leistungsfähiger DSP-Bords die Auswahl des richtigen DSPs von nicht zu unterschätzender Bedeutung. Die folgenden Erläuterungen zu den Prozessorstrukturen sollen Ihnen in Zusammenhang mit den technischen Parametern die Beurteilung der Leistungsfähigkeit der digitalen Signalprozessoren für Ihren Anwendungsfall erleichtern und somit Voraussetzungen für eine optimale Auswahl schaffen.
Ausgegangen wird hierbei von allgemeinen Kriterien zur Leistungseinschätzung von DSPs. Im weiteren werden die bekanntesten DSPs – vom ersten 16-Bit-Festkomma-DSP bis zu den modernen 32-Bit-Gleitkomma-DSPs – vorgestellt.

Eigenschaften und Strukturen

Die Vielfalt der Algorithmen in der digitalen Signalverarbeitung, insbesondere die große Anzahl von Filterfunktionen und Transformationen, erfordert neben der unterschiedlichen Kombination arithmetischer Befehle einen algorithmenspezifischen Datenzugriff, wie er zum Beispiel vom Butterfly-Algorithmus der schnellen Fourier-Transformation (FFT) bekannt ist. Daraus ergeben sich unterschiedliche Anforderungen an die internen Strukturen und Schnittstellen der DSPs.
Um eine maximale Leistungsfähigkeit zu erreichen, werden für einige oft anzuwendende Algorithmen speziell angepaßte Prozessoren entwickelt. Sie besitzen eine für einen konkreten Algorithmus optimierte Struktur, beispielsweise der Datenflußprozessor μPD 7281 von NEC (vgl./1/), der FIR-Prozessor IMS A100 von Inmos (vgl. /2/) oder der FFT-Prozessor 34161 von Zoran (vgl. /3/). Diese stark anwendungsorientierten DSPs können oft nur für eine spezielle Anwendung oder eine Gruppe von Anwendungsfällen eingesetzt werden. Sie sind im allgemeinen keine Stand-alone-Prozessoren und benötigen deshalb einen oder mehrere Prozessoren, die die Datenbereitstellung für den DSP und die Steuerung des Systems übernehmen.
Unvierselle DSPs orientieren sich im Gegensatz zu den oben erwähnten DSPs auf allgemeine Anforderungen in der Signalverarbeitung. Sie besitzen eine Struktur, die die Programmierung unterschiedlichster Algorithmen ermöglicht und einen maximalen Datendurchsatz der Arithmetikeinheit unterstützt. Dazu sind unter anderem notwendig:
– leistungsfähige arithmetische Funktionsgruppen, insbesondere Multiplizierer
– spezielle Register und Rechenwerke zur Adreß- und Datenmanipulation ohne Nutzung des Akkumulators (ACC) oder der Arithmetik- und Logikeinheit (ALU)
– eine interne bzw. vollständige Harvard-Struktur.
In Analogie zu den Mikroprozessoren besitzen diese DSPs im allgemeinen eine vollständige Busschnittstelle für den Anschluß externer Speicher und die Adressierung von Ein-/Ausgabeschnittstellen. Deshalb können diese DSPs größtenteils auch als Stand-alone-Prozessoren eingesetzt werden. Der Einsatz in Multiprozessorsystemen wird oft unterstützt.

Harvard-Struktur

Wie schon erwähnt, hat die Prozessorstruktur einen wesentlichen Einfluß auf die Leistungsfähigkeit des DSPs. Die meisten Mikroprozessoren haben eine *Von-Neumann-Architektur*. Sie ist gekennzeichnet durch eine sequentielle Abarbeitung aller Operationen, bedingt durch einen gemeinsamen Adreßraum für Daten und Befehle. Dies ermöglicht unter anderem einen geringeren internen und externen Hardware-Aufwand (Bild 1a; x,y,z-Speicher; R1, R2-Register). Der für die Signalverarbeitung erforderliche hohe Datendurchsatz kann erst durch eine Trennung der Befehlsadressierung von der Adressierung der Daten erreicht werden. Dementsprechend befinden sich bei der *Harvard-Architektur* die Daten und die Befehle in unterschiedlichen Adreßräumen, um ein paralleles Lesen von Befehlen und Daten zu ermöglichen. Neben den getrennten Speichern erfordert dies eine doppelte Busstruktur (Daten- und Programmbus) und entsprechende Adreßregister und -rechenwerke. Durch paralleles Laden der Daten und Befehle kann der Datendurchsatz bei gleicher Rechenleistung der ALU nahezu verdoppelt werden (Bild 1b).
Eine weit höhere Auslastung der ALU und damit eine Erhöhung der Rechenleistung wird erreicht, wenn die externen Busschnittstellen der Prozessoren an die der ALU angepaßt werden. Ausgehend von den drei Datenschnittstellen einer ALU erfordert ein maximaler Datendurchsatz die Bereitstellung der Daten und der Befehle über vier separate Busse: einen Befehlsbus, zwei Dateneingangs- und einen Datenausgangsbus mit entsprechenden Adreßgeneratoren (Bild 2).
Eine Modifizierung der eigentlichen Harvard-Struktur mit einem Buskoppler zwischen den „Datenbussen" beider Adreßräume oder einem austauschbaren Speicherblock ermöglicht eine höhere Flexibilität bei der Programmabarbeitung und ebenfalls eine paral-

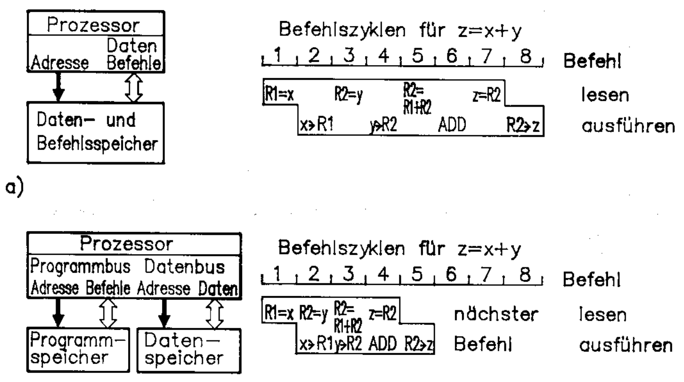


Bild 1 Vergleich von Harvard- und Von-Neumann-Struktur
a) Von-Neumann-Strukur
b) Harvard-Struktur mit getrenntem Adreß- und Datenbus

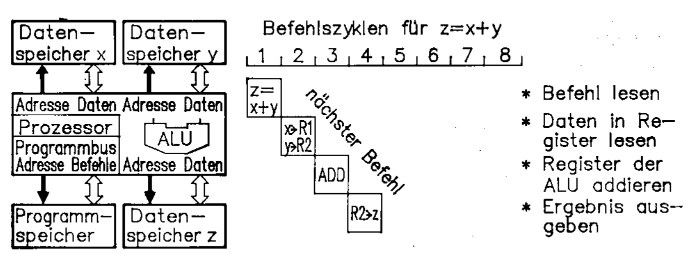


Bild 2 Erweiterte Harvard-Struktur mit 3 Datenbussen

lele Bereitstellung der Operanden für die ALU; in diesem Fall über den Daten- und den Programmbus.

Leistungsbestimmende Parameter

Leistungsbestimmende Parameter für DSPs, an denen sich das enorme Entwicklungstempo der letzten Jahre erkennen läßt, sind:
– *kurze Befehlszykluszeiten* (insbesondere für Multiplikation). Mit dem Wechsel der Technologie von NMOS auf CMOS und mit der Erhöhung der Integrationsdichte verringerten sich die Befehlszykluszeit von zirka 400 ns auf weniger als 100 ns und die Verlustleistung von einigen Watt auf wenige 100 mW.
– *Wortbreite und internes Rechenformat*. Neben den DSPs mit einem 16-Bit-Festkommaformat werden seit Ende der 80er Jahre zunehmend DSPs mit einem 32-Bit-Gleitkommaformat angeboten.
– *Adressierbarer Speicher*. Der Adreßraum von einigen KByte erhöht sich mit den 32-Bit-DSPs auf einige MByte.
– *Schnittstellen*. Neben parallelen E/A-Schnittstellen besitzen die modernen DSPs serielle Schnittstellen und sind interrupt-, multiprozessor- und DMA-fähig.
– *Befehlssatz*. Komfortable Befehle ermöglichen vielfältige Adressierungsarten. Wiederholbare Befehle und Blocktransferbefehle verringern die Befehlsausführungszeiten durch Ausnutzen interner Pipeline-Strukturen und Adreßrechenwerke. Saldierende Akkumulatorbefehle unterstützen die Implementierung zahlreicher Algorithmen der Signalverarbeitung.
Die Einteilung der DSP-Typen erfolgt nach firmenspezifischen Gesichtspunkten anhand der Entwicklungsetappen der jeweiligen Hersteller. So werden vergleichbare Gleitkommaprozessoren als DSPs der zweiten oder vierten Generation angeboten. Ausgehend von den obigen Parametern, insbesondere bezüglich der Leistungsfähigkeit der arithmetischen Funktionsgruppen, soll hier nur zwischen zwei verschiedenen Leistungsklassen unterschieden werden. Die untere Leistungsklasse der DSPs besitzt eine 16-Bit-Festkommaarithmetik, und die darauffolgende ist durch eine 32-Bit-Gleitkommaarithmetik gekennzeichnet. Anhand typischer Vertreter will ich im folgenden die Strukturen und Funktionsweisen digitaler Signalprozessoren erläutern.

Intel 2920 als erster Signalprozessor

Mit dem Schaltkreis **i2920** stellte Intel 1979 den ersten programmierbaren Signalprozessor (Bild 3) /4/ für die Sprachverarbeitung vor. Die Integration der analogen Umsetzer und des digitalen Verarbeitungsteils auf einem Chip reduzieren den externen Schaltungsaufwand auf ein Minimum.
Die AD-Umsetzung erfolgt mittels eines Abtast-Haltegliedes (Sample-and-Hold-Glied), eines Komparators und des 8-Bit-DA-Umsetzers durch sukzessive Approximation.
Für die digitale Verarbeitung mit einer Datenbreite von 25 Bit stehen ein Dual-Port-RAM (40 × 25 Bit), ein Schieberegister und eine ALU zur Verfügung. Die 25-Bit-ALU ermöglicht Addition, Subtraktion, Betragsbildung sowie logische UND- und Exklusiv-ODER-Operationen von zwei Operanden des Dual-Port-RAMs.
Gesteuert wird der Signalprozessor durch die Dekodierung der 24 Bit breiten Befehlswörter des Programmspeichers, der über einen rücksetzbaren Zähler adressiert wird. Programmverzweigungen sind bei diesem Signalprozessor noch nicht möglich. Die Aufspaltung der Befehlswörter in einzelne Felder für die ALU, den Dual-Port-Speicher, das Schieberegister und den Analogteil ermöglicht eine parallele Steuerung der einzelnen Funktionsgruppen. So kann beim i2920 die digitale Signalverarbeitung parallel zur AD- bzw. DA-Umsetzung erfolgen.
Die funktionell geschlossene Lösung von Befehlsgenerierung, digitaler Verarbeitung und analoger Ein-/Ausgabe ermöglicht einerseits den Aufbau minimaler Hardwarekonfigurationen, begrenzt aber andererseits die Leistungsfähigkeit der ALU bei großen Datenmengen und die Universalität des Prozessors durch das Fehlen digitaler Schnittstellen. Die Leistungsfähigkeit des ersten Signalprozessors war begrenzt durch:
– die Befehlszykluszeit von 400 ns
– die sequentielle Befehlsabarbeitung ohne Programmverzweigungen
– die geringe Eingangsdatenrate, bedingt durch das Iterationsverfahren des AD-Umsetzers
– das Fehlen eines schnellen Multiplizierers.
Damit war der Einsatz des i2920 im wesentlichen auf die niederfrequente, analoge Signalverarbeitung begrenzt. Die Forderung nach wesentlich schnelleren Signalprozessoren führte zur Entwicklung digitaler Signalprozessoren mit integrierten Multiplizierern und schnellen digitalen E/A-Schnittstellen ohne langsame, analoge Funktionsgruppen.

DSPs mit Festkommaarithmetik

Die Vertreter dieser Leistungsklasse sind charakterisiert durch
– eine 16-Bit-Festkommaarithmetik
– Adreßräume von bis zu 64 K Wörtern.
Der **μPD7720** von NEC /4/ war der erste digitale Signalprozessor mit einem 16-Bit-Multiplizierer. Die Ausführung einer Multiplikation innerhalb eines Befehlszyklus (250 ns) führte zu einer wesentlich höheren arithmetischen Leistungsfähigkeit gegenüber den Mikroprozessoren (Bild 4).
Neben der internen Harvard-Struktur ermöglichen zusätzliche Datenbusse zum Hauptdatenbus einen hohen Grad an Parallelität bezüglich der gleichzeitigen Ausführung verschiedener arithmetischer Operationen und Speicherzugriffe. Außerdem ermöglicht der arithmetische Verarbeitungsteil ein Pipelining bei der Befehlsausführung. So werden die Ergebnisse des Multiplizierers (31 Bit) über einen speziellen, aber nur 16 Bit breiten Bus an die 16-Bit-ALU weitergeleitet. Die zur ALU gehörenden zwei 16-Bit-Akkumulatoren (ACC-A, ACC-B) ermöglichen arithmetische Operationen mit einfacher Genauigkeit (16 Bit) und doppelter Genauigkeit (32 Bit).
Der im Programmfestwertspeicher abgelegte Mikrobefehlskode von 23 Bit steuert den μPD7720. Gegenüber der Befehlsabarbeitung beim i2970 ermöglichen jedoch der Stack und die Interruptsteuerung eine wesentlich flexiblere Programmierung. Der Datenaustausch des μPD7720 mit der Peripherie ist nur über die bidirektionale 8-Bit-Ein-/Ausgabeschnittstelle oder über die serielle Schnittstelle (vollduplex) möglich. Das Fehlen einer 32-Bit-ALU und eines 32-Bit-Akkumulators einschließlich der entsprechenden 32-Bit-Busse und einer 16 Bit breiten Busschnittstelle bedeuten eine Leistungsbegrenzung dieses DSPs.
Beim **TMS32010** von Texas Instruments (vgl. /5/) treten diese leistungsbegrenzenden Differenzen in der Struktur nicht auf. Die Daten, die Befehle und acht auswählbare parallele Ein-/Ausgabeschnittstellen besitzen die gleiche Datenbreite von 16 Bit. Die arithmetische Leistungsfähigkeit des TMS32010 wird neben seiner internen Harvard-Struktur vor

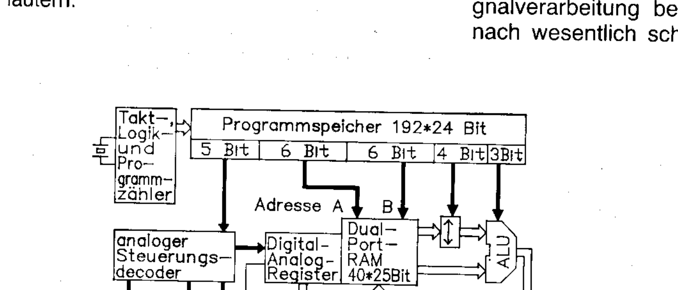


Bild 3 Funktionelles Blockschaltbild des I2920

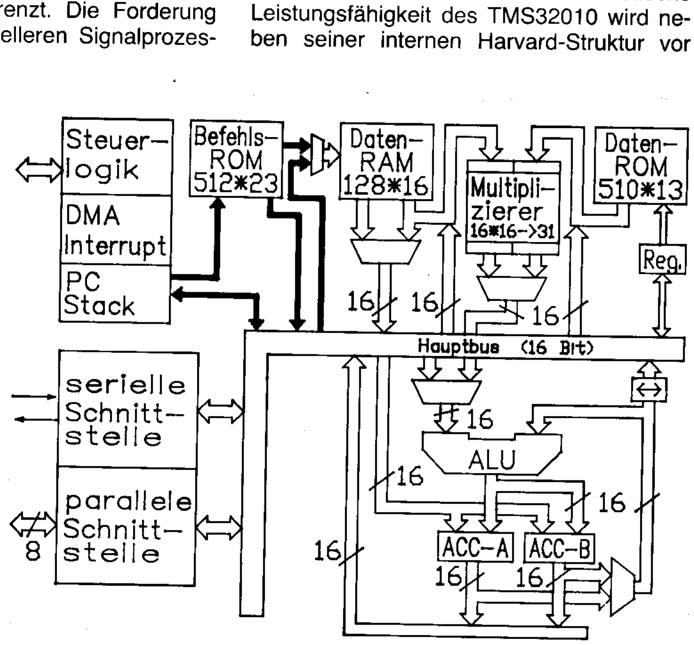


Bild 4 Blockschaltbild des μPD7720

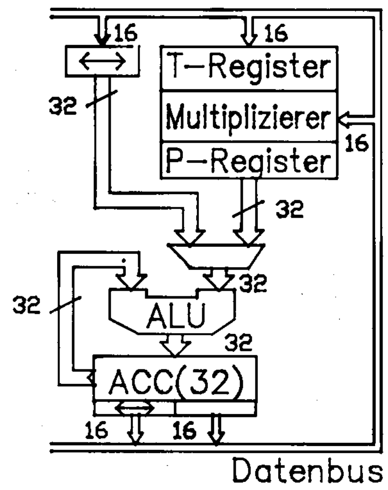


Bild 5 Arithmetikteil des TMS32010

allem durch die konsequente Pipeline-Struktur des Arithmetikteils erreicht.
Das Aufeinanderfolgen von Multiplizierer (16 × 16 → 32 Bit), 32-Bit-ALU und 32-Bit-Akkumulator, einschließlich der Schieberegister, ermöglichen eine leistungsfähige Pipeline-Verarbeitung (Bild 5). Charakteristisch für diesen und alle weiteren DSPs ist, daß die Verarbeitungsgenauigkeit der ALU und des Akkumulators dem Produkt des Multiplizierers entspricht.
Durch eine Kopplung des Daten- und des Programmbusses ist ein Datenaustausch zwischen beiden Adreßräumen möglich. Neben der Modifizierung des Programms in Abhängigkeit von den Daten ermöglicht dies auch das schnelle Einlesen von Daten über den Programmspeicher.
Die universellen Eigenschaften des TMS32010 führten zu einer weiten Verbreitung und zur Entwicklung weiterer Versionen dieses Prozessortyps.
Die ersten DSPs (μPD7720, TMS32010, MB8764, PC85010) aber auch modernere DSPs, wie der TMS320C25, besitzen zusätzlich zum internen RAM einen internen Programmfestwertspeicher. Je nach Version werden diese DSPs mit maskenprogrammierbarem Speicher (ROM) oder mit EPROM ausgeliefert. Man unterscheidet zwischen zwei Betriebsarten. Im *Mikroprozessormodus* ist dieser Speicher ausgeblendet und somit neben dem internen RAM der gesamte externe Programmspeicher adressierbar. Im *Mikrocomputermodus* wird der Adreßraum des On-chip-Programm-ROMs in den unteren Teil des externen Adreßraumes eingeblendet. Der On-chip-ROM kann dann für die Bereitstellung von Startroutinen, Unterprogrammen oder des vollständigen Programms genutzt werden und entsprechende Laderoutinen verkürzen oder erübrigen. Dies ist insbesondere für solche Anwendungen sinnvoll, die keine Busschnittstelle besitzen.
Hauptnachteile der ersten DSPs sind einerseits die kleinen Adreßräume der Daten- und Programmspeicher, die meist nur eine eingeschränkte Adressierung externer Speicher ermöglichten, und andererseits die Befehlszykluszeit von 200 ns und mehr. Mit der weiteren Entwicklung der Schaltkreistechnologien erfolgten bis zur Mitte der achtziger Jahre vor allem die Verbesserung einzelner Parameter und die Erweiterung vorhandener Strukturen, wie
– die Befehlszykluszeit
– die Größe des Adreßraumes und des internen Speichers
– die Anzahl der Register und der Adreßrechenwerke
– die externen Schnittstellen,
ohne daß jedoch die Struktur des Arithmetikteils wesentlich verändert wurde. Als Beispiel für diese Entwicklung kann die DSP-Familie TMS 320 von Texas Instruments mit den Prozessoren TMS32010, TMS32020 und TMS320C25 angesehen werden.
Trotz Beibehaltung der Struktur des TMS32010, insbesondere des arithmetischen Teils mit dem 16-Bit-Multiplizierer, der 32-Bit-ALU und dem Akkumulator, konnte die Leistungsfähigkeit durch Erweiterung bestehender Funktionsgruppen und Schnittstellen wesentlich erhöht werden. Dazu gehören beim **TMS32020** (/6/, /7/)
– die Erweiterung des On-chip-RAMs, der Bestandteil der auf jeweils 64 K Wörter erweiterten Adreßräume ist
– die Multiprozessorfähigkeit
– die bidirektionalen seriellen Schnittstellen.
Eine kompatible CMOS-Version zum TMS32020 von TI ist der U 320C20 vom Forschungszentrum Mikroelektronik Dresden (ZMD). Dieser im Bild 6 dargestellte DSP hat jedoch im Unterschied zum TMS einen wesentlichen geringeren Stromverbrauch und eine höhere Taktrate (vgl. /8/). Die ersten nennenswerten Stückzahlen werden noch in diesem Jahr erwartet.
Der **TMS320C25** ist eine weiterentwickelte, weitgehend kompatible CMOS-Version des TMS32020. Durch die Verdopplung der Taktfrequenz auf 40 MHz wird eine Befehlszykluszeit von 100 ns erreicht. Die Weiterentwicklung des TMS320C25 gegenüber seinem Vorgänger bezieht sich vor allem auf die Erweiterung vorhandener Funktionsblöcke /9/. Dazu gehören die Vergrößerung des Kellerspeichers und der Hilfsregister des Hilfsregisterrechenwerkes, das Hinzufügen eines On-chip-Programm-ROMs, eines Überlaufbits (carry flag), eines Adreßzählers, eines zweiten Datenspeichers und eines Sende- und eines Empfangsschieberegisters für das Puffern der Daten der seriellen Schnittstelle sowie die Erweiterung des Befehlssatzes um 24 Befehle zur Untersützung von
– Arithmetik mit erweiterter Genauigkeit
– Akkumulator-, Registeroperationen und Bitmanipulationen
– der Programmierung von Filteralgorithmen und der Radix-2-Adressierung für die FFT (bit reversed indexed addressing).
Eine weitere Leistungssteigerung wird durch den Concurrent-DMA-Modus erreicht. Er bewirkt, daß die Programmabarbeitung im Hold-Modus des DSPs fortgesetzt wird, sofern nur interne Speicheroperationen ausgeführt werden.
Weitere DSPs dieser Leistungsklasse sind
– der TS68931 von Thomson Semiconductor (vgl. /10/)
– der DSP56000 von Motorola (vgl. /11/)
– der PC85011 von Philips (vgl. /12/).
Diese Prozessoren besitzen ebenfalls eine interne Harvard-Struktur, deren getrennte interne Daten- und Programmadreßräume über eine gemeinsame externe Busschnittstelle verfügen, so daß externe abgespeicherte Befehle und Daten nur nacheinander mit langsameren externen Speicherzugriffsbefehlen geladen werden können. Über die Busschnittstelle erfolgt ebenfalls die Adressierung der Ein-/Ausgabeschnittstellen.
Eine weitere Gruppe von DSPs mit Festkommaarithmetik besitzt eine vollständige Harvard-Struktur. Als Vertreter sei der speicherlose DSP **ADSP2100** von Analog Devices /13/ genannt. Ein wesentlicher Vorteil seiner Struktur ist, daß der parallele Speicherzugriff für Daten und Befehle nicht auf die On-chip-Speicher begrenzt ist. Dies wird durch die separate Herausführung der Daten- und Befehlsbusse einschließlich der dazugehörigen Adreßbusse erreicht (Bild 7). Dieser DSP sollte vor allem dann eingesetzt werden, wenn beispielsweise aus Gründen der Anzahl der Parameter oder der Art des Datenzugriffs keine wesentlichen Vorteile durch die Nutzung eines schnellen internen RAMs

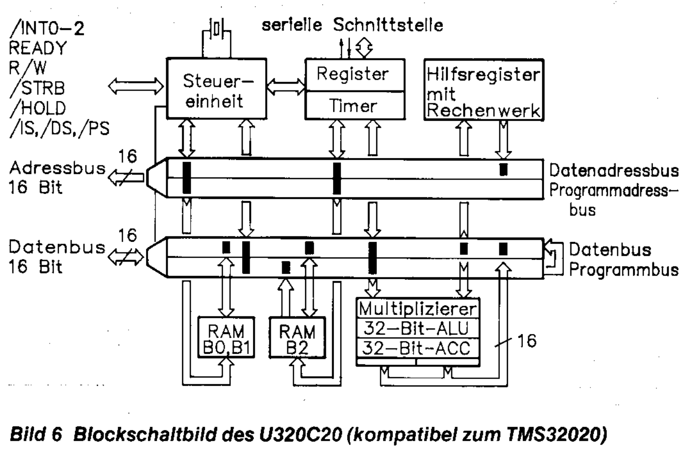


Bild 6 Blockschaltbild des U320C20 (kompatibel zum TMS32020)

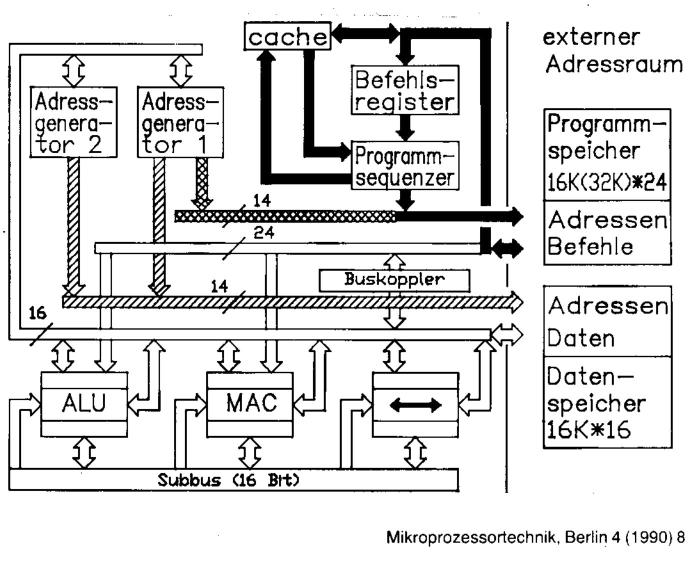


Bild 7 Blockschaltbild des ADSP2100

gegenüber dem externen RAM erreicht werden können. Neben den getrennten Bussystemen für Daten- und Programmspeicher ermöglicht ein interner Transportbus den Datentransfer zwischen der ALU, den MACs (16-Bit-Multiplizierer und 40-Bit-Akkumulator) und einem Schieberegister. Zwei unabhängige Adreßgeneratoren können den Daten- und Programmspeicher adressieren, wodurch eine parallele Datenzuführung aus beiden externen Speichern erfolgen kann. Die Befehlsgenerierung muß in diesem Fall über den Programmsequenzer in Verbindung mit dem Cache-Speicher oder durch einen Repeat-Befehl erfolgen. Spezielle Register ermöglichen eine zyklische Adressierung im Datenspeicher zur Unterstützung der Implementierung digitaler Filteralgorithmen.

Eine erweiterte Harvard-Struktur besitzt der **LM32900** von National Semiconductor (Bild 8). Sie ist gekennzeichnet durch zwei 16-Bit-Datenbusse, einen 28 Bit breiten Befehlsbus und entsprechend herausgeführte Adreßbusse mit einem jeweiligen Adreßraum von 64 K Wörtern. Die sich aus der Anzahl der Busse ergebenden 172 Anschlüsse deuten den notwendigen externen Aufwand beim Einsatz dieses DSPs an. Die Befehlszykluszeit beträgt 100 ns. Daraus ergeben sich maximale Zugriffszeiten von nur 25 ns für die externen Datenspeicher.

Hervorzuheben ist weiterhin das vollständig statische CMOS-Design, was beliebig niedrige Taktfrequenzen bei direkt proportionalem Leistungsabfall gestattet /14/ und damit eine taktsynchrone Arbeit mit der Peripherie und ein taktgesteuertes Anhalten des DSPs ermöglicht.

Im Unterschied zu den Mikroprozessoren besitzen die 16-Bit-DSPs keine Adreßlogik zur Erweiterung des Adreßraumes, so daß dieser auf je 64 K Wörter begrenzt ist. Des weiteren begrenzt die 16-Bit-Festkommaarithmetik die Rechengenauigkeit für einige Anwendungen derart, daß Gleitkommaoperationen unumgänglich sind. Viele DSPs unterstützen deshalb die Programmierung von Gleitkommaoperationen durch spezielle Befehle. Die programmtechnische Gleitkommaberechnung erfordert jedoch, einschließlich entsprechender Fehlerbehandlungsroutinen, wesentlich mehr Befehlszyklen (40 und mehr) als die entsprechenden Festkommaoperationen, so daß die zeitlichen Anforderungen ohne weiteres überschritten werden können. In solch einem Fall ist der Einsatz eines DSPs mit Gleitkommaarithmetik unumgänglich.

DSPs mit Gleitkommaarithmetik

Diese DSPs ermöglichen Gleitkommaoperationen innerhalb eines Befehlszyklus mit einer Datenbreite von 24 oder 32 Bit. Sie haben im allgemeinen eine Zykluszeit von unter 150 ns und einen Adreßraum von über 1 MByte. Digitale Signalprozessoren und Gleitkommamultiplizierer besitzen jedoch oft firmenspezifische Formate. So hat der HD 62810 von Hitachi eine 16-Bit-Mantisse und einen Exponenten mit 4 Bit Breite (16E4). Ein standardisiertes Gleitkommaformat ist das nach IEEE 754 (Tafel 1).

Der erste DSP mit einer 32-Bit-Gleitkommaarithmetik ist der **μPD77230**. Er besitzt ein dem IEEE-754-Standard ähnliches Gleitkommaformat, wobei der Befehlssatz entsprechende Befehle für die Transformierung zum Standard enthält. Eine entsprechende Verarbeitung von Daten im Festkommaformat mit einer Datenbreite der Mantisse sind ebenfalls möglich. Das Bild 9 zeigt das Blockschaltbild des μPD77230 mit dem Multiplizierfeld, der Steuereinheit, der Prozeßeinheit und den integrierten Speichern /15/.

Der kleine Adreßraum dieses Gleitkomma-DSPs von 4 K (bzw. 8 K) × 32 Bit für die Adressierung externer Befehle und Daten ist ein wesentlicher Nachteil. Der arithmetische Verarbeitungsteil besteht aus einem 32-Bit-Gleitkommamultiplizierer und einer Prozeßeinheit (PU – processing unit) mit einer internen Verarbeitungsbreite von 55 Bit. Der Eingangsmultiplexer der Prozeßeinheit ermöglicht die Auswahl von verschiedenen Datenquellen für die dreistufige Pipeline, die sich aus dem Schieberegister, der ALU und 8 Arbeitsregistern – WR0...WR7 – zusammensetzt.

Hervorzuheben ist die unterschiedliche Schnittstellenkonfigurierung im Master- und im Slave-Modus, die im linken unteren Teil des Blockschaltbildes dargestellt ist. Sie unterstützt den Aufbau von Multiprozessorstrukturen mit Master- und Slave-Prozessoren. Im Master-Modus besitzt der DSP ein vollständiges externes 32-Bit-Speicherinterface mit 13 Adreßleitungen. Demgegenüber erfolgt im Slave-Modus eine Neuzuordnung der Pins in der Art, daß einem Host eine CPU-, ein I/O-Interface und ein lokales Speicherinterface zur Verfügung steht. Aufgrund der Befehlswortbreite von 32 Bit kann im Slave-Modus keine Adressierung eines externen Programmadreßraumes erfolgen, da der 32-Bit-Datenbus zu einem 8-Bit-Datenbus, einem 16-Bit-Host-Interface und weiteren vier I/O-Ports umgeschaltet wird.

Weitere 32-Bit-Gleitkommaprozessoren sind
- der DSP32 von AT&T (vgl. /16/)
- der 86232 von Fujitsu (vgl. /17/)
- der DSP96002 von Motorola (vgl. /18/)
- der TMS320C30 von TI (vgl. /19/).

Der **DSP32** hat im Gegensatz zu den meisten DSPs keine Harvard-, sondern eine Von-Neumann-Struktur. Um trotz der einfacheren Busstruktur einen hohen Datendurchsatz zu erreichen, besitzt dieser DSP einen Hochgeschwindigkeitsbus für Daten und Befehle. Die On-chip-Speicher ermöglichen bis zu vier Speicherzugriffe je Befehlszyklus (80 ns).

Der **TMS320C30** gehört zur dritten Generation der DSP-Familie von Texas Instruments. Er hat eine Befehlszykluszeit von nur 60 ns. In seiner Struktur unterscheidet er sich wesentlich von seinen Vorgängern. Mehrere parallele Daten- und Adreßbusse, drei parallele Speicherblöcke und ein Cache-Speicher ermöglichen eine umfangreiche Parallelisierung beim Datentransfer zwischen den Speichern, der Peripherie und der CPU. Das ist eine Voraussetzung für die effiziente Auslastung der einzelnen Funktionsblöcke des DSPs (Bild 10). Die drei Hauptbusse (Programm-, Daten- und DMA-Bus) können wahlweise über die Systembusschnittstelle auf den Adreßraum von 16 M × 32 Bit zugreifen. Mit der internen Trennung der Adreßräume besitzt der TMS320C30 eine Harvard-Struktur, wobei keine eindeutige Zuordnung der Speicher zu den einzelnen Bussen (Adreßräumen) existiert. Dies erübrigt den Buskoppler zwischen dem Daten- und dem Programmbus, wie er uns von den 16-Bit-DSPs her bekannt ist.

Eine wesentliche Entlastung der CPU erfolgt über die Organisation des Datentransportes durch den DMA-Controller. Dieser ermöglicht die Bedienung der Schnittstellen über den Peripherie-Bus unabhängig von der Programmabarbeitung durch die CPU. An diesen Bus sind eine zweite Busschnittstelle mit einem auf 8 K × 32 Bit reduzierten Adreßraum, zwei serielle Schnittstellen und zwei Timer angeschlossen.

Tafel 1 Gleitkommaformat nach IEEE 754

einfache Genauigkeit	31 V 1	30 32 Exponent 8	22 0 Mantisse 23
doppelte Genauigkeit	63 V 1	62 52 Exponent 11	51 0 Mantisse 52

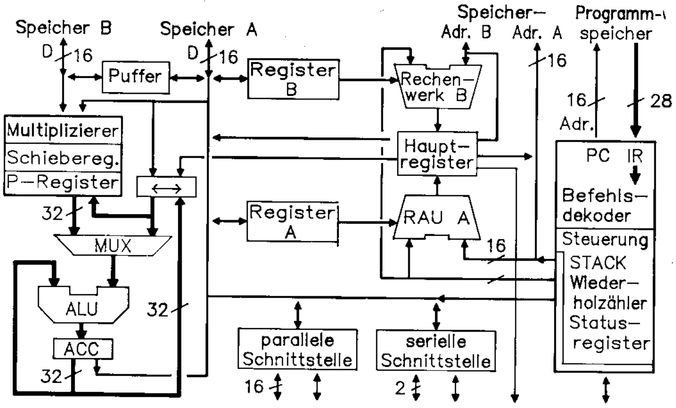


Bild 8 Blockschaltbild des LM 32900

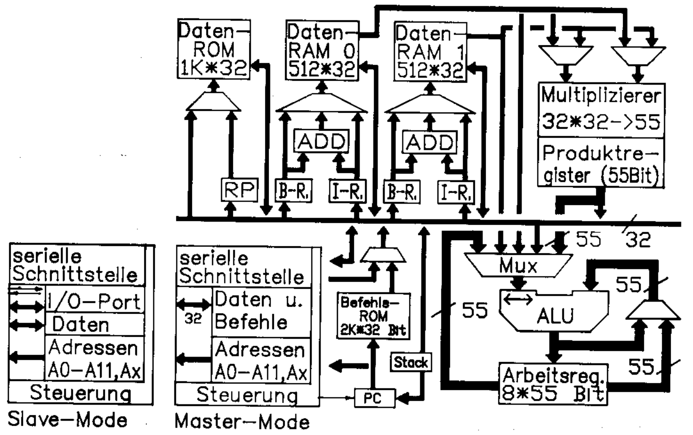


Bild 9 Blockschaltbild des μPD77230

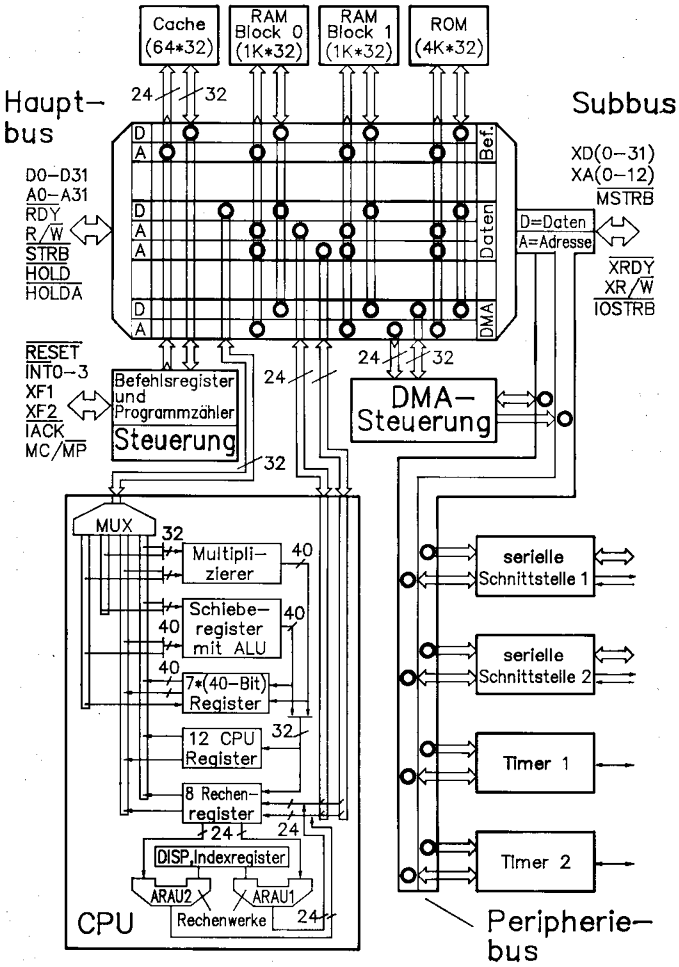


Bild 10 Blockschaltbild des TMS 320C30

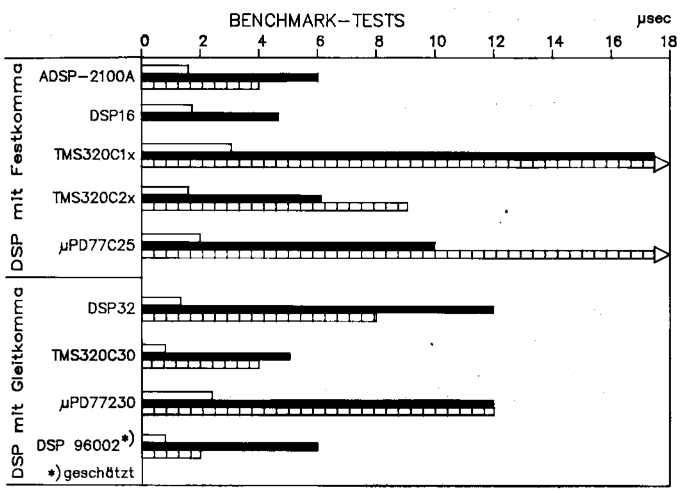


Bild 11 Benchmark-Tests für einige DSPs (/21/)
a) Matrixmultiplikation (3 × 3) (3 × 1) vom Speicher in einen Speicher, wobei die Ausgangsmatrix zerstört wird
b) FIR-Filter (finite impulse response filter) mit 67 Schritten über eine parallele Schnittstelle
c) komplexe Radix-2-FFT (1024 Werte) mit Sortierung, wobei die Ausgangsdaten zerstört werden

Signalprozessorkarte DSP20 für den EC 1835

Die Komplettierung eines EC 1835, IBM PC/AT oder Kompatiblen mit einer oder mehreren Signalprozessorsteckkarten DSP20 ermöglicht den Aufbau eines leistungsfähigen Multirechnersystems zur digitalen Signalverarbeitung. Gegenüber dem in Personalcomputer optional vorhandenen Arithmetikprozessor ermöglicht ein DSP20-Signalrechner die schnelle Ausführung umfangreicher arithmetikintensiver Signalverarbeitungsalgorithmen parallel zum Hostrechner (PC).
Für die Konfiguration eines Multisignalrechnersystems wurde im Zentrum für wissenschaftlichen Gerätebau der Akademie der Wissenschaften die Globalspeichersteckkarte **GSK20** entwickelt, die eine Wechselspeicherbank mit je 64 K Wörtern statischen Speicher besitzt. Für die analoge zweikanalige Meßdatenerfassung wurde die Steckkarte **AMK20** mit dem AD-Umsetzer AD 9003 (Analog Devices) entwickelt, die in Verbindung mit der GSK20 die zweikanalige Meßwerterfassung bis zu einer Samplingfrequenz von 1 MHz ermöglicht, während für Samplingfrequenzen ≦ 70 kHz die Meßdaten direkt vom Personalcomputer übernommen werden können.
Für die komfortable Programmentwicklung wird eine integrierte Entwicklungsumgebung bereitgestellt, die Editor, Makefunktion, Assembler, Linker, Debugger und Monitor unter einer einheitlichen Oberfläche anbietet. Gegenüber dem Arithmetikprozessor können umfangreiche Algorithmen (z. B. Vektor- und Matrixoperationen) parallel zum Host berechnet werden. Die Algorithmen sind über Treiber von Turbo-Pascal 5.0 aufrufbar.

DSP-Arbeitsplatz für U 320C20

Im Geraer Betrieb von Carl Zeiss JENA wurde in Zusammenarbeit mit dem Institut für Automatisierung und dem Zentralinstitut für Kybernetik und Informationsprozesse der Akademie der Wissenschaften der Arbeitsplatz für digitale Signalprozessoren DSP-AP zur Erarbeitung und zum Testen der Software für die Signalprozessoren U 320C20 und TMS 32020 entwickelt.
Der DSP-AP unterstützt die Simulation von Programmen, die Softwareinbetriebnahme unter Echtzeitbedingungen und die Inbetriebnahme und Prüfung anwenderspezifischer Baugruppen. Der Arbeitsplatz besteht aus: einem PC (XT/AT) als Bedien- und Crossrechner und einem Beistellgefäß, das unter anderem 9 Steckplätze mit MMS-16-Bus und/oder Signalprozessoren-Bus, einen Einplatinenrechner auf der Basis des MMS 16 als Koppeleinheit mit seriellem Interface zum Bedienrechner, einen Signalprozessorentwicklungsmodul sowie eine Breaklogik enthält. Zum Signalprozessorentwicklungsmodul gehört ein Signalprozessor U 320C20, ein 8-K-Wörter-Programm- und ein 16-K-Wörter-Datenspeicher, eine schnelle Treiberlogik mit Statusdekoder und Statusport.
Der Nutzer hat die Möglichkeit, seine eigenen Baugruppen, beispielsweise ADC, DAC, Rechner oder weitere externe Speicher an den DSP-Modul anzuschließen. Es stehen dem Anwender drei Steckplätze mit dem Signalprozessor-Bus und dem MMS-16-Bus zur Verfügung. Zur Schonung der Bus-Steckverbinder sind diese Steckplätze mit leicht auswechselbaren Zwischenadaptern ausgestattet. An diesem Bus sind Adreß-, Daten- und Steuersignale getrieben. Die Stromversorgung besitzt dazu auch ausreichende Leistungsreserven für den Anwender.
Zum DSP-AP werden folgende Softwarekomponenten zur Verfügung gestellt: Cross-Assembler und Linker, Simulator, Editor, Hardware-Check und Laufzeitdebugger.
Der Simulator bietet menügesteuerte Möglichkeiten, die wesentlichsten Hardwareelemente des DSPs zu jedem beliebigen Zeitpunkt zu kontrollieren und gegebenenfalls zu verändern. Ein komfortabler Service zur Steuerung der Programmabarbeitung über Breakpoints und Interrupts und zur Modellierung der Ein- und Ausgabefunktionen des DSPs zeichnen dieses Testhilfsmittel aus. Es bietet weiterhin einen vom Nutzer beeinflußbaren Mechanismus zur Ermittlung realer Abarbeitungszeiten von Programmen unter einer konkreten Hardware.
Der Laufzeitdebugger DESPERADO bietet unter anderem die Optionen: schrittweise Programmabarbeitung bis zu verschiedenen Befehlen oder Zuständen, Ausrührung eines abgewiesenen (aborted) Interrupts, Unterbrechung der laufenden Debug-Funktion, Zurücksetzen des Signalprozessors sowie Füllen des Programm- bzw. Datenspeichers. Die interaktive Bedienung des Debuggers erfolgt menügesteuert und unter Nutzung von Fenstertechnik.

Die CPU besitzt ebenfalls eine parallele Busstruktur für parallele Adressierung und Datenbereitstellung für die ALU und den Multiplizierer. Der 32-Bit-Parallelmultiplizierer verarbeitet Daten im 16- und im 32-Bit-Festkomma- oder Gleitkommaformat innerhalb eines Befehlszyklus. In Verbindung mit den erweiterten Datenregistern (40 Bit) können die Produkte des Multiplizierers im 40-Bit-Gleitkommaformat (32E8) gespeichert oder in der ALU weiterverarbeitet werden. Die Adressierung der Daten kann parallel über die beiden Adreßrechenwerke und die dazugehörigen Adreßregister erfolgen.
Mit einer CPU-Leistung von 16,7 MIPS und 33 MFLOPS gehört der TMS320C30 zu den leistungsfähigsten DSPs. So benötigt er für eine 1024-Punkte-FFT (Radix-2) 3,04 ms /20/.

Neben der Entwicklung von neuen, immer leistungsfähigeren DSPs erfolgt ebenfalls ein Redesign von NMOS- in CMOS-DSPs und die Weiterentwicklung von 16-Bit-Festkomma-DSPs. Diese DSPs sind weitgehend kompatibel zu ihren Vorgängern, haben einen wesentlich geringeren Leistungsverbrauch und kürzere Befehlszykluszeiten. Der niedrigere Preis gegenüber 32-Bit-DSPs,

Tafel 2 Technische Parameter ausgewählter DSPs

Hersteller Typ	Technologie	Zykluszeit (ns)	Datentyp Datenbreite	interner Speicher*)		externer Adreßraum		externe Busse		CPU			Steuerung		I/O	
				Daten	Programm	Daten	Programm	Daten	Programm	Multiplizierer	ALU	ACC	Interrupt	DMA	parallel	seriell
Analog Devices ADSP2100A	CMOS	80	Festkomma 16	/	/	16 K	32 K × 24	1	1	16 × 16 → 40			4	ja	nein	nein
AT&T DSP16	CMOS	55	Festkomma 16	512	2 K ROM	64 K	/	/	1	16 × 16 → 32		36	ja		2	1
DSP32C	CMOS	80	Gleitkomma 32 (24E8)	4 K ROM oder 512 und 2 × 512		16 M		1		24E8 × 24E8 → 32E8		40	ja	ja	ja	1
Motorola DSP56000	CMOS	97.5	Festkomma 24	512 512 ROM	2 K ROM	128 K	64 K	1		24 × 24 → 56	56	56	4	ja	ja	2
DSP96002	CMOS	75	Gleitkomma 32 (24E8)	2 × 512	512 32 ROM	2 × 1 G		2		32E11 × 32E11 → 96 24E8 × 24E8			4	ja	/	/
National Semiconduktor LM32900	CMOS	100	Festkomma 16	/	/	128 K	64 K × 28	2	1	16 × 16 → 32	32	32	ja		ja	ja
NEC µPD7720	NMOS CMOS	250 200	Festkomma 16	128 512×13R	512 × 23R	/	/	/	/	16 × 16 → 31	16	16	ja	ja	ja	ja
µPD77230	CMOS	150	Gleitkomma 32 (24E8)	512 1 K ROM	2 K ROM	8 K	4 K	1		24E8 × 24E8 → 47E8	55	55	2		ja	ja
Philips/ Segnetics PC85010	CMOS	250	Festkomma 16	2 × 128 512 ROM	987 × 40R	/	/	/	/	16 × 16 → 32	40	40	ja		1	2
TI TMS32010 TMS320C10	NMOS CMOS	200	Festkomma 16	144	1536 ROM	/	4 K	/	1	16 × 16 → 32	32	32	ja	/	ja	/
TMS32020	NMOS	200	Festkomma 16	288 oder 544	256 oder 0	64 K	64 K	1		16 × 16 → 32	32	32	3	ja	ja	1
TMS320C25	CMOS	100	Festkomma 16	288/544	256/0 4 K ROM	64 K	64 K	1		16 × 16 → 32	32	32	3	ja	ja	1
TMS320C30	CMOS	60	Gleitkomma 32 (24E8)	2 × 1 K	Cache 4 K ROM	16 M		2		32E8 × 32E8 → 40	40	40	4	ja		2

*) Speichertyp RAM, wenn nicht als ROM gekennzeichnet (in Byte)

verbesserte Parameter und die bereits vorliegenden Einsatzerfahrungen seitens der Anwender ermöglichen eine schnelle Produktüberarbeitung und damit die Entwicklung kostengünstiger Produkte.
Als Beispiele seien hier der µPD77C20, TMS320C10, der TMS320E17-25 sowie der TMS320C50 genannt. Allen gemeinsam ist der wesentlich geringere Leistungsverbrauch und eine kürzere Befehlszykluszeit. Des weiteren besitzt der TMS320E17 einen vergrößerten On-chip-ROM, und er wurde gegenüber seinem Vorgänger (TMS32010) um zwei serielle Schnittstellen und eine Koprozessor-Schnittstelle erweitert. Der von TI angekündigte TMS320C50 ist ein 16-Bit-DSP mit einer Festkommaarithmetik, dessen Busstruktur der des TMS 320C30 ähnelt. Er besitzt einen On-chip-Speicher von 10,5 K Wörtern und eine *Parallel Logic Unit* (PLU), die schnelle Bitmanipulationen über einen Speicherbereich ermöglicht. Mit einer Befehlszykuszeit von 50 ns wird er zu den schnellsten 16-Bit-DSPs gehören.

Vergleich ausgewählter DSPs

Der direkte Vergleich von Prozessoren, insbesondere von DSPs, ausschließlich anhand der technischen Parameter ist nur bedingt sinnvoll. Die digitalen Signalprozessoren unterscheiden sich in ihrer Struktur, den technischen Parametern und dem Befehlssatz oft so stark voneinander, daß eine Leistungseinschätzung im Zusammenhang mit dem konkreten Einsatzfall vorgenommen werden muß. Dabei sollten die für den Algorithmus notwendige Rechengenauigkeit, der Speicherplatzbedarf, der Daten- und Programmspeicherzugriff, Prozessorschnittstellen, der Befehlssatz und die verfügbaren Entwicklungswerkszeuge ebenso beachtet werden wie die Befehlszykluszeit. Um diese Abschätzungen zu erleichtern, sind im Bild 11 einige Benchmark-Tests und in Tafel 2 (vgl. /21/, /22/) die Zusammenfassung technischer Parameter ausgewählter DSPs aufgeführt.
Bei den 16-Bit-DSPs besitzt Texas Instruments mit der Produktfamilie TMS320 den größten Marktanteil. Aufgrund der Einsatzvorbereitungen für den U 320C20 gibt es in den unterschiedlichsten Betrieben und Einrichtungen der DDR umfangreiche Erfahrungen beim Einsatz des TMS32020 und des TMS320C25. Informationen über Entwicklungssysteme oder Boards sind unter anderem über die Nutzergemeinschaft (ZKI der AdW, Prof. Albrecht) oder beim Hersteller (ZMD) zu erfahren.

Literatur

/1/ Hell, R.: Der „Image Pipeline Prozessor" von NEC. Elektronikschau, Wien 64 (1988) 6, S. 26
/2/ Datenblatt A100. Inmos, 1988
/3/ A new class of chip. electronics, July 1986, S. 58
/4/ Quarmby, D.: Signalprozessoren: 2920, 7720 und 320. München/Wien: Verlag Oldenbourg 1986
/5/ TMS 32010 User's Guide. Texas Instruments, 1983
/6/ Kropp, G.: Signalprozessor bietet Multiprozessorfähigkeit. Elektronik, München 35 (1986) 6, S. 53
/7/ Datenblatt TMS32020. Texas Instruments, 1985
/8/ Autorenkollektiv, Heuer, G.: Handbuch U 320C20. Berlin: Verlag Technik, in Vorbereitung
/9/ Second-Generation TMS 320 User's Guide. Texas Instruments, 1988
/10/ Bursky, D.: Digital signal processor runs complex calculations in amere 160 ns. Electronic Design, (1987) Nov. 14, S. 99E
/11/ Anderer, G.: Digitaler Signalprozessor mit RISC-Architektur. Elektronik, München 35 (1986) 10, S. 210
/12/ Picha, R.: Signalprozessor-IC's der Familie SP 50. Elektronikschau, Wien 63 (1987) 4, S. 46
/13/ Binder, N.: DSP-Chip mit hoher Rechenleistung. Elektronikschau, Wien 62 (1986) 11, S. 38
/14/ Annecke, K. H.: LM 32900 – Schneller Signalprozessor für große Daten- und Programmspeicher. elektronik industrie, München 18 (1987) 2, S. 30
/15/ Fleischer, Ch.; Geyer, M.: Floatingpoint-Signalprozessor. Elektronikschau, Wien 63 (1987) 11, S. 40
/16/ Wright, M: CMOS DSP IC offers 80 ns cycle time – operates on IEEE floating-point numbers. EDN (1987) Sept. 3, S. 99
/17/ Cushman, R. H.: µP-like DSP chips. EDN (1983) Sept., S. 186
/18/ Straub, A.: 96-Bit-Register und Hochsprachen – DSP-Chip dringt in Hochleistungsbereich vor. Elektronik, München 37 (1988) 19, S. 82
/19/ Simar, R.; Hames, M.: CMOS DSP chips packs punch of super-computer. Elektronic Disign (1987) March 19, S. 103
/20/ Papamichalis, P.; Simar, R.: The TMS 320C30 Floating-Point Digital Signal Processor. IEEE Micro (1988) Dez., S. 13
/21/ Titus, J.: Special report „DSP IC's". EDN, (1986) Okt., S. 162
/22/ Von Bechen, P.: Integrierte Digitale Signalprozessoren. Elektronik, München 37 (1988) 19, S. 145
/23/ Shear, D.: EDN's DSP Benchmarks. EDN, (1988) Sept. 29, S. 126

✉ KONTAKT ☎

Akademie der Wissenschaften der DDR, Zentralinstitut für Kybernetik und Informationsprozesse, Rudower Chaussee 5, Berlin 1199; Tel. 6 74 24 97

Real-Zahlen

Zeichnung: Steger

Peripherieschaltkreise der Reihen DS 8600 und DL 8600

Elke Badermann, Dr. Dieter Kühnel, Siegfried Löbbicke, Peter Piontek, Frankfurt (Oder)

DMA-Logikschaltkreis DS 8601 DC

Der DMA-Logikschaltkreis DS 8601 DC ist ein Schottky-TTL-Schaltkreis, der für den Einsatz in DMA-organisierten Interfacegeräten vorgesehen ist. Er liefert die Logik zur Ausführung von Handshake-Operationen bei Ausnutzung der Steuerung des Bussystems. Hat sich die Busherrschaft der DMA durchgesetzt, erzeugt sie die erforderlichen Steuersignale zur Ausführung von Lese-, Schreib- oder Multiplex-Transfers entsprechend einer Einstelltabelle.
Der DS 8601 besitzt einen Steuereingang, welcher im einen Fall beliebig viele Zyklen und im anderen Fall nur vier Zyklen gestattet, um die Busherrschaft wieder abzugeben. Er liefert an den TTL-Ausgängen mit Gegentakt-Endstufe 8 mA und an den Opencollector-Ausgängen 70 mA Treiberstrom bei Low-Pegel.
Der DS 8601 DC wird in einem 20poligen Dual-in-line-Plastgehäuse geliefert.

Anschlußbelegung, Schaltzeichen

Bild 1 zeigt die Anschlußbelegung des DS 8601 DC mit einer kurzen Beschreibung in Tafel 1. Bild 2 zeigt das Schaltzeichen des DS 8601 DC.

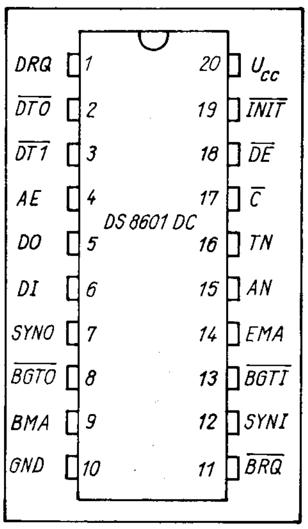


Bild 1 Anschlußbelegung des DS 8601 DC

Tafel 1 Beschreibung der Anschlüsse des DS 8601 DC (E – Eingang; A – Ausgang)

Bezeichnung	Anschluß	Typ	Name und Funktion
DRQ	1	E	DATA-REQUEST (DMA-Anforderung) Ein High an diesem Eingang leitet die Forderung einer Übertragung an den Bus ein. Ein Low beendet die Übertragung und somit die Busherrschaft.
$\overline{\mathrm{DT0}}$	2	E	DATA IN/OUT (Datenübertragung-Ein-/Ausgabe) Mit diesem Signal findet die Auswahl der Transferart entsprechend einer Wahrheitstabelle statt. Während eines DI/D0-Transfers muß dieses Signal umkippen, um den Übertragungszyklus zu beenden. Bei offenem Eingang nimmt dieser Anschluß High ein. Einstelltabelle: $\overline{\mathrm{DT1}}$ / $\overline{\mathrm{DT0}}$ / Transfer X / L / DI/D0 L / H / DI H / H / D0
$\overline{\mathrm{DT1}}$	3	E	DATA IN (Datenübertragung – Eingabe) Mit diesem Signal findet die Auswahl der Transferart entsprechend der oben angegebenen Einstelltabelle statt.
AE	4	A	ADDRESS ENABLE (Adreßfreigabe) Dieses Signal zeigt an, daß eine Adresse auf den Bus gebracht werden kann.
DO	5	A	DATA OUT (Datenausgabe/Master) Dieses Signal zeigt an, daß an den Ausgängen des Busmasters gültige Daten anliegen. → Schreibfreigabe für den Speicher
DI	6	A	DATA IN (Dateneingabe/Master) Dieses Signal zeigt an, daß der Busmaster bereit ist, Daten zu empfangen. → Lesefreigabe des Speichers
SYNO	7	A	TRANSMIT SYNCHRONIZE (Synchronisation/Ausgang) Dieses Signal entsteht in der DMA als Hinweis, daß ein Transfer ausgeführt wird.
$\overline{\mathrm{BGTO}}$	8	A	DMA GRANT OUTPUT (Busgewährung/Ausgang; open collector) Dieses Signal ist das verzögerte Signal von $\overline{\mathrm{BGTI}}$, solange keine Anforderung (DRQ = Low) vorhanden ist. Ansonsten ist dieser Ausgang nicht aktiv. Er kann direkt an den Bus angeschlossen sein.
BMA	9	A	MASTER (Busherrschaft) Ein High an diesem Ausgang zeigt an, daß die DMA die Busherrschaft besitzt und ein Transfer ausgeführt wird.
GND	10		Masse
$\overline{\mathrm{BRQ}}$	11	A	BUS-REQUEST (Busanforderung; open collector) Ein Low an diesem Anschluß weist darauf hin, daß die DMA die Busherrschaft anfordert. Dieser Ausgang kann direkt an den Bus angeschlossen sein.
SYNI	12	E	RECEIVE SYNCHRONIZE (Synchronisation/Eingang) Bei diesem Signal wird die DMA zum Master entsprechend folgender Beziehung: $\overline{\mathrm{SYNI}}$ x $\overline{\mathrm{AN}}$ X BMA ENA = BMA. BMA ENA ist ein internes Signal der DMA-Logik.
$\overline{\mathrm{BGTI}}$	13	E	DMA GRANT INPUT (Busgewährung/Eingang) Bei einem Low an diesem Eingang setzt sich die Busherrschaft durch, wenn die Busanforderung noch vorhanden ist (DRQ = H). Ansonsten wird dieses Signal verzögert zum Ausgang $\overline{\mathrm{BGTO}}$ übertragen.
EMA	14	E/A	TIME-OUT (Eingang: Freigabe $\overline{\mathrm{BRQ}}$ oder Ausgang: Freigabe BMA; open collector) Als Ausgangsvariante ist dieser Anschluß Low bei BMA ENA = H. Bei BMA ENA = L geht der Ausgang in den gesperrten Zustand über. Wird am Eingang ein L angelegt, wird $\overline{\mathrm{BRQ}}$ = L verhindert. Wird am Eingang ein H angelegt, ist $\overline{\mathrm{BRQ}}$ wirksam, falls $\overline{\mathrm{BRQ}}$ wegen der Begrenzung auf max. 4 Übertragungen rückgesetzt wurde. An diesem Anschluß kann zur Verzögerung von $\overline{\mathrm{BRQ}}$ ein RC-Netzwerk verwendet werden.
AN	15	E	REPLY (Antwort) Dieses Signal dient zum Freigeben oder Sperren des freien Taktimpulses entsprechend der Einstelltabelle. Auch mit diesem Signal kann die DMA Master werden, wenn folgende Beziehung gilt: $\overline{\mathrm{SYNI}}$ x $\overline{\mathrm{AN}}$ x BMA ENA = BMA
TN	16	E	COUNT FOUR (Übertragungsanzahl) Ein H am Eingang gestattet, ein Maximum von vier Zyklen zu übertragen, bevor wieder die Busherrschaft abgegeben wird. Ein L hebt diesen Vorgang auf, und es kann ein unbegrenzt langer Transfer stattfinden, solange DRQ = H einnimmt. Bei offenem Eingang nimmt dieser Anschluß High an.
$\overline{\mathrm{C}}$	17	E	CLOCK (Takt) Dieses Taktsignal wird zum Erzeugen aller Transfers benutzt.
$\overline{\mathrm{DE}}$	18	A	DATA ENABLE (Datenfreigabe) Dieses Signal zeigt an, daß Daten auf den Bus gebracht werden können.
$\overline{\mathrm{INIT}}$	19	E	INITIALIZE (Initialisierung) Dieses Signal stellt den Anfangszustand der DMA her, bevor DRQ = H mit einer Busanforderung beginnen kann. Ist $\overline{\mathrm{INIT}}$ = L, sind folgende Signale unwirksam: $\overline{\mathrm{BRQ}}$, BMA, $\overline{\mathrm{DE}}$, AE, SYNO, DI, DO
U_{cc}	20		Betriebsspannung

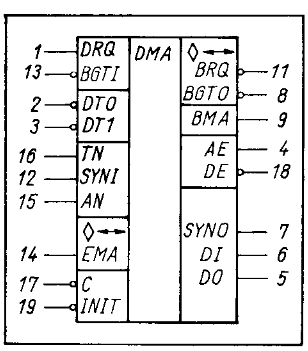


Bild 2 Schaltzeichen des DS 8601 DC (positive Logik)

Funktionsweise

Bild 3 stellt den Ablauf einer einmaligen DI/D0-Übertragung mit Herstellung des Ausgangszustandes durch $\overline{\mathrm{INIT}}$ = L und mit dem taktunabhängigen Handshake zu Beginn der Übertragung dar. Entsprechend der Belegung der Eingänge DT0, DT1, TN und DRQ werden die drei Übertragungsarten und ihre Übertragungslänge realisiert.

Serien-Parallel-/Parallel-Serien-Wandler-Schaltkreis DS 8609 DC

Der DS 8609 DC ermöglicht den Datenaustausch zwischen seriellem und parallelem Datenformat. Er wird in Schottky-TTL-Tech-

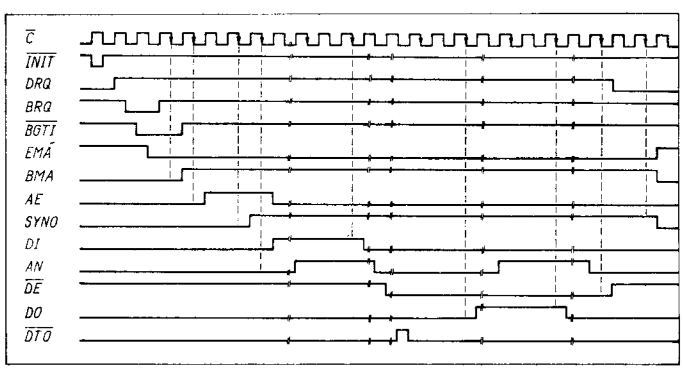


Bild 3 Ablauf der DI/D0-Übertragung

Tafel 2 Anschlußbeschreibung des DS 8609 DC

Be-zeich-nung	An-schluß	Typ	Name und Funktion
GNDT	1	–	GROUND TTL Masseanschluß für alle internen TTL-Stufen (Logik, Flipflops); extern mit Anschluß 24 verbinden!
$\overline{SYND}$	2	A	SYNCHRONIZATION DETECTED Wenn das Synchronwort erkannt wird, liegt ein Low an diesem Ausgang. Das Signal ist nur für Testzwecke gedacht.
C	3	E	CLOCK Schiebetakt für alle Register und Takt für die meisten Status-Flipflops
OFUF	4	A	OVERFLOW/UNDERFLOW Dieses Signal kann nur durch einen Rücksetzimpuls an Anschluß 23 auf Low gesetzt werden. Bei der Serien-Parallel-Umsetzung wird es aktiviert (High), wenn innerhalb eines Wortzyklus keine Rückmeldung des Datenempfangs an Anschluß 6 erfolgt ist. Bei der Parallel-Serien-Umsetzung wird es aktiviert, wenn innerhalb eines Wortzyklus kein Ladetakt-Impuls an Anschluß 26 vorliegt
RDYP	5	A	READY PARALLEL Ein High an diesem Ausgang zeigt an, daß das parallele Ausgabewort stabil anliegt (bei Serien-Parallel-Umsetzung) oder die Bereitschaft zum Empfang paralleler Daten vorhanden ist (bei Parallel-Serien-Umsetzung).
DAK	6	E	DATA ACKNOWLEDGE Dieses Signal wirkt nur bei der Serien-Parallel-Umsetzung. Mit einem High an DAK wird die Abnahme des parallelen Datenwortes bestätigt.
D	7–14 27–34	E/A	PARALLEL DATA IN/OUT paralleles bidirektionales Datentor
DS1 DS0	15 16	E E	DATA INPUT SERIAL serielle Dateneingänge
DSC	17	E	DATA INPUT SERIAL CONTROL Dieses Signal bestimmt, welcher der beiden seriellen Dateneingänge mit dem Schieberegister verbunden ist. Ein High an DSC aktiviert DS1; ein Low aktiviert DS0.
$\overline{OEP}$	18	E	OUTPUT ENABLE PARALLEL Ein High an diesem Anschluß bringt alle 16-D-Ausgänge in den Tristate-Zustand.
ECT	19	E	ENABLE COUNTER Bei Serien-Parallel-Umsetzung erlaubt ein High an diesem Anschluß die Erkennung des Synchronmusters zum anschließenden Start des Bitratenzählers. Bei Parallel-Serien-Umsetzung wird der Bitratenzähler sofort freigegeben, wenn ein High anliegt.
GNDE	20	–	GROUND ECL Masse für interne Spannungsreferenzen und ECL-/EFL-Stufen (Bei Spezialanwendung keine externe Verbindung mit Anschluß 1 und 24)
WLS0 WLS1	21 22	E E	WORD LENGTH SELECT Diese beiden Anschlüsse bestimmen die Länge des parallelen Ein- und Ausgabewortes. Im folgenden ist die Codierung für die möglichen Wortlängen angegeben: | 16 Bit | 10 Bit | 8 Bit WLS0 | H | H | L WLS1 | H | L | H Gleichzeitiges Low an beiden Steuereingängen führt zur Fehlfunktion und ist daher unzulässig.
$\overline{R}$	23	E	RESET Ein Low-Impuls an diesem Anschluß setzt alle Register, Zähler und die meisten Status-Flipflops zurück.
GNDP	24	–	GROUND PARALLEL Masse für die Ausgangstreiber des parallelen Tores
MSP	25	E	MODE SERIAL/PARALLEL Dieses Signal definiert die grundlegende Arbeitsweise des Schaltkreises. High bedeutet Serien-Parallel-Umsetzung, Low Parallel-Serien-Umsetzung.
$\overline{LP}$	26	E	LOAD PARALLEL Dieses Signal wirkt nur bei Parallel-Serien-Umsetzung. Mit der L-H-Flanke eines Low-Impulses an diesem Eingang wird das an D anliegende parallele Wort in ein internes Register geladen.
$\overline{OES}$	35	E	OUTPUT ENABLE SERIAL Ein High an diesem Eingang bringt die beiden seriellen Ausgänge ZDOS (Anschluß 38) und ZCOS (Anschluß 39) in den Tristate-Zustand.
CW	36	A	WORD CLOCK Der Wortratentakt wird im Schaltkreis in Abhängigkeit von der gewählten Wortlänge erzeugt. Seine Periodendauer entspricht n Schiebetaktimpulsen (n: eingestellte Wortlänge); das Tastverhältnis beträgt etwa 1:1
DOS	37	A	DATA OUTPUT SERIAL serieller Datenausgang (ohne Tristate-Verhalten)
ZDOS	38	A	TRISTATE DATA OUTPUT SERIAL serieller Datenausgang (mit Tristate-verhalten
ZCOS	39	A	TRISTATE CLOCK OUTPUT SERIAL Dieser Ausgang gibt – zeitlich verzögert – das an C (Anschluß 3) anliegende Taktsignal aus. Die Verzögerung dient der genauen Synchronisation des Taktes auf die an ZDOS (Anschluß 38) ausgegebenen seriellen Daten. Das Tristate-Verhalten wird, ebenso wie das von ZDOS, durch $\overline{OES}$ (Anschluß 35) gesteuert.
U_{cc}	40	–	Betriebsspannung

nik hergestellt. Der Schaltkreis kann sowohl eine Serien-Parallel- als auch eine Parallel-Serien-Umsetzung vornehmen. Dabei ist er auf eine Wortlänge von 8, 10 oder 16 Bit einstellbar. Für den Datentransfer stehen ein bidirektionales paralleles Tor (16 Bit) sowie zwei serielle Ausgänge und zwei serielle Eingänge zur Verfügung. Das parallele Tor und eines der seriellen Ausgangstore besitzen Tristate-Ausgänge, die über separate Enable-Signale angesteuert werden. Der zweite serielle Ausgang hat eine normale (Totempole-)Endstufe. Von den seriellen Eingängen kann je einer über ein spezielles Steuersignal aktiviert werden.
Bild 4 zeigt ein Blockschaltbild, welches die wesentlichen inneren Schaltungskomplexe des Schaltkreises enthält. Wie daraus zu ersehen ist, enthält der Schaltkreis zwei 16-Bit-Registerketten. Diese werden abwechselnd mit dem parallelen Tor und den internen Steuersignalen verbunden, um einen lückenlosen Datentransfer in beiden Funktionsarten zu gewährleisten. Der Datentransfer kann bis zu einer Taktfrequenz von 28 MHz vorgenommen werden. Eine Beschreibung der Anschlüsse enthält Tafel 2.

Serien-Parallel-Umsetzung

In dieser Funktionsart sucht der Schaltkreis in einem seriellen Datenstrom zunächst ein Synchronmuster, das den Beginn der zu übertragenden Daten kennzeichnet. Zuvor ist der Schaltkreis mit einem Reset-Impuls am Steuereingang $\overline{R}$ in seinen definierten Anfangszustand zu bringen. Dieser Zustand bleibt erhalten, bis das Synchronmuster erkannt wird. Danach werden die Daten entsprechend der eingestellten Wortlänge quantisiert und an den parallelen Ausgängen zur Verfügung gestellt.
Durch ein Signal am Ausgang RDYP meldet der Schaltkreis, daß ein Datenwort stabil an den parallelen Ausgängen zur Verfügung steht. Während dieses Wort in einer Registerkette gespeichert bleibt, wird in die zweite Registerkette das nächste Wort seriell eingelesen. Drei Taktimpulse vor dem Abschluß dieser Operation muß am Eingang DAK eine Empfangsbestätigung für das parallele Wort eingegangen sein. Bleibt diese aus, so wird am Ausgang OFUF ein Übertragungsfehler gemeldet. Während die Verarbeitung der Daten fortgesetzt wird, kann die Fehlermeldung erst mit dem nächsten generellen Reset gelöscht werden.
Die seriell eingelesenen Daten erscheinen zeitverzögert wieder an den seriellen Ausgängen DOS und ZDOS. Diese Verzögerung beträgt für eine Wortlänge von
8 Bit: 21 Taktimpulse
10 Bit: 25 Taktimpulse
16 Bit: 37 Taktimpulse.

Parallel-Serien-Umsetzung

Nach einem Reset-Impuls am Eingang $\overline{R}$ befindet sich der Schaltkreis im Anfangszustand. Es erfolgt keine Suche nach einem Synchronmuster, so daß der Bitratenzähler sofort gestartet wird.
Durch einen Ladeimpuls am Eingang $\overline{LP}$ wird das parallele Wort in eine der beiden Registerketten gespeichert, während der Inhalt der anderen Registerkette mit dem Takt C an die seriellen Ausgänge geschoben wird.
Ähnlich wie bei der Funktion Serien-Parallel-Umsetzung erfolgt am Ausgang OFUF die Anzeige eines Übertragungsfehlers, wenn der Ladeimpuls im dafür vorgesehenen Zeitraum ausbleibt.

Bidirektionaler Bustreiber DS 8638 DC

Der Schaltkreis DS 8638 DC enthält 8 bidirektionale Bustreiber in Schottky-TTL-Technik. Die 8 Bustreiber dienen einer asynchronen Zweiwege-Kommunikation zwischen Open-collector- und Tristate-Bussen. Es werden Daten von einem A-Bus (opencollector) zu einem B-Bus (tristate) oder von einem B-Bus zu einem A-Bus in Abhängigkeit vom Pegel am Richtungssteuereingang (DIR) übertragen. Der Enable-Eingang ($\overline{CS}$) kann zum Trennen der beiden Busse voneinander benutzt werden.
Die maximale Treiberstromstärke beträgt 70 mA an den A-Ausgängen und 25 mA an den B-Ausgängen. Die Funktionsweise des Schaltkreises verdeutlicht Tafel 3.

Busempfänger-Schaltkreis DL 8640 DC

Die integrierte Schaltung DL 8640 DC enthält vier Busempfänger in Low-power-Schottky-TTL-Technik mit je zwei NOR-verknüpften Eingängen mit einer typischen Schaltschwelle von 1,5 V. Die Gegentaktausgangsstufen haben eine Treiberfähigkeit von 16 mA bei $U_{OL} \leq 0{,}4$ V. Der Schaltkreis ist für den Einsatz in busorganisierten Datenübertragungssystemen vorgesehen, deren 120-Ohm-Datenbus' durch 180-Ohm-Widerstände vom Bus zur Betriebsspannung und

Tafel 3 Funktionstabelle des Bustreibers DS 8638 DC

Steuereingänge $\overline{CS}$	DIR	Operation
L	L	$\overline{B}$-Daten zum A-Bus
L	H	$\overline{A}$-Daten zum B-Bus
H	X	Bustrennung

H ≙ High-Pegel; L ≙ Low-Pegel; X ≙ beliebig

Tafel 4 Funktionstabelle des Busempfängers DL 8640 DC

Eingänge A	B	Ausgänge Y
L	L	H
L	H	L
H	L	L
H	H	L

H ≙ High-Pegel; L ≙ Low-Pegel

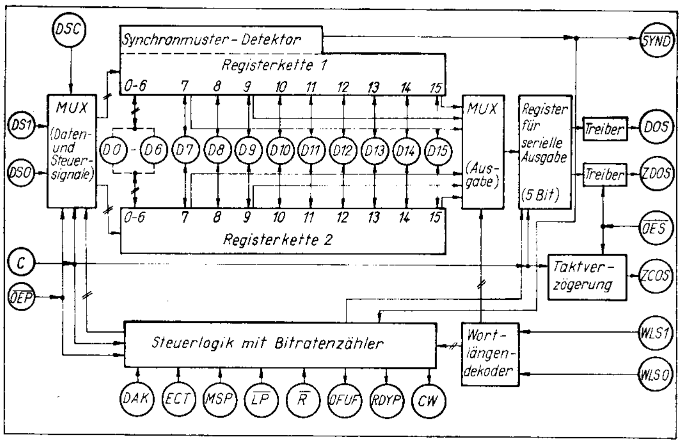


Bild 4 Blockschaltbild des DS 8609 DC

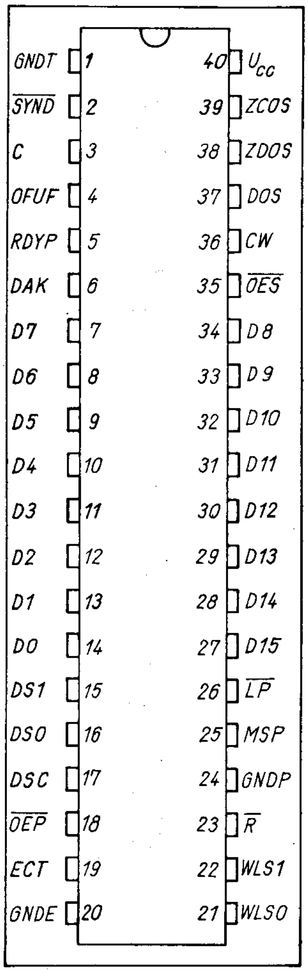


Bild 5 Anschlußbelegung des DS 8609 DC

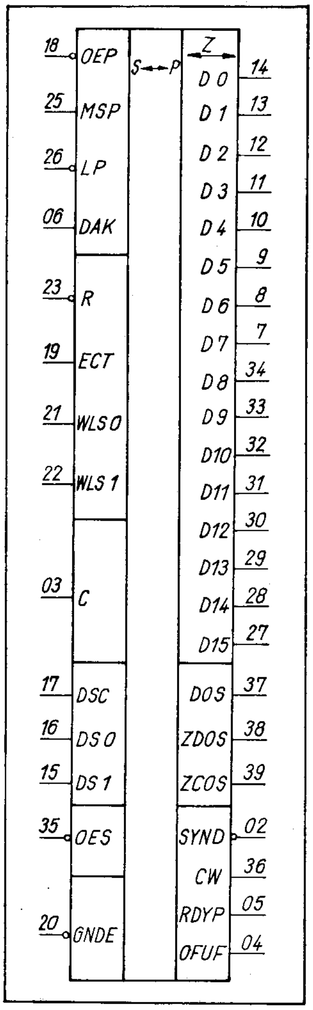


Bild 6 Schaltzeichen des DS 8609 DC (positive Logik)

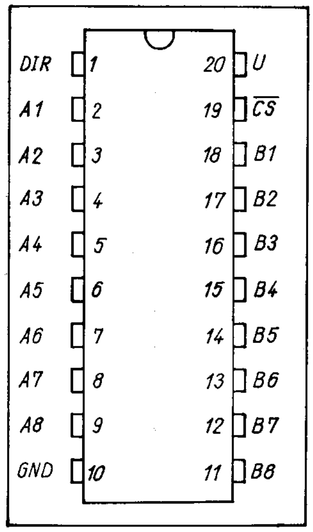


Bild 7 Anschlußbelegung des DS 8638 DC
A1...A8 Ein-/Ausgänge für Daten (open collector)
B1...B8 Ein-/Ausgänge für Daten (tristate)
CS chip select (output enable)
DIR direction (Richtungssteuerung)
U_{cc} Betriebsspannung
GND Masse

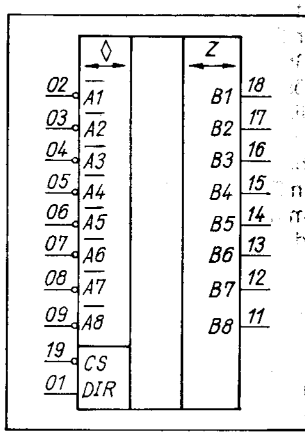


Bild 8 Schaltzeichen des DS 8638 DC (positive Logik)

Bild 11 Anschlußbelegung des DL 8641 DC ▶
A1...A4 Dateneingänge
B1...B4 Datenbus (bidirektional)
Y1...Y4 Ausgänge
G1, G2 Disable-Eingänge
U_{cc} Betriebsspannung
GND Masse

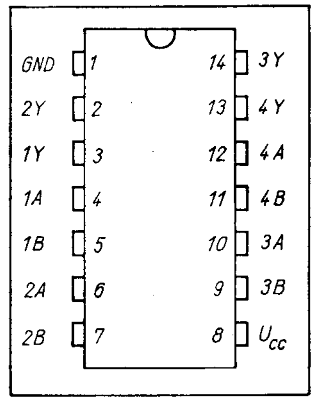


Bild 9 Anschlußbelegung des DL 8640 DC
1A...4A Dateneingänge
1B...4B Dateneingänge
1Y...4Y Ausgänge
U_{cc} Betriebsspannung
GND Masse

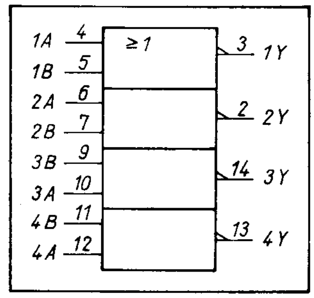
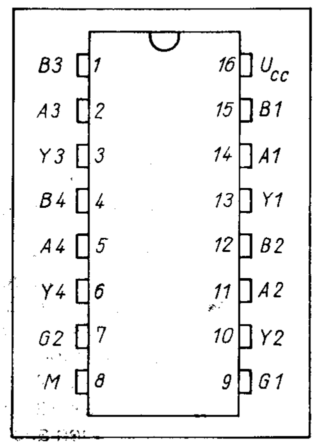


Bild 10 Schaltzeichen des DL 8640 DC (positive Logik)



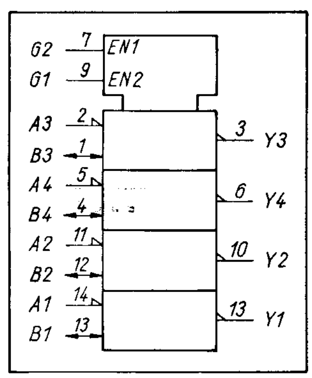


Bild 12 Schaltzeichen des DL 8641 DC (positive Logik)

durch 390-Ohm-Widerstände vom Bus nach Masse abgeschlossen ist.

Busempfänger- und Treiber-Schaltkreis DL 8641 DC

Die integrierte Schaltung DL 8641 DC enthält vier Busempfänger-/Treiber in Low-power-Schottky-TTL-Technik und ist für den Einsatz in busorganisierten Datenübertragungssystemen vorgesehen, deren 120-Ohm-Datenbus durch 180-Ohm-Widerstände vom Bus zur Betriebsspannung und durch 390-Ohm-Widerstände vom Bus nach Masse abgeschlossen ist.
Die Bus-Ausgänge sind Open-collektor-Ausgänge mit einer Treiberfähigkeit von 50 mA bei $U_{OL} \leq 0{,}7$ V. Die Busempfängereingänge haben eine typische Schaltschwelle von 1,5 V, die Empfängerausgänge sind Gegentaktausgangsstufen mit einer Treiberfähigkeit von 16 mA bei $U_{OL} \leq 0{,}4$ V.
Mit den NOR-verknüpften Disable-Eingängen G1 und G2 wird die Funktion des Schaltkreises gesteuert (entsprechend Tafel 5).

Tafel 5 Funktionstabelle des Busempfängers/Treibers DL 8641 DC

Funktion	Disable-eingang G1	G2	Treiber-eingang A	Treiberausgang/ Empfängereingang B	Empfänger-ausgang Y
Empfänger	H	X	X	H	L
	H	X	X	L	H
	X	H	X	H	L
	X	H	X	L	H
Treiber	L	L	H	L	H
	L	L	L	H	L

H ≙ High-Pegel; L ≙ Low-Pegel; X ≙ beliebig

✉ KONTAKT ☎

VEB Halbleiterwerk Frankfurt (Oder), Postfach 379, Frankfurt (Oder), 1200; Tel. 46 32 75

Rasterbilder auf dem PC

Dr. Peter Döring, Berlin

Grafische Darstellungen sind eine anschauliche Ausgabeform von Programmergebnissen. Sie gestatten es, wesentliche Eigenschaften schnell zu erfassen und erleichtern die Analyse der Ergebnisse bedeutend. Andererseits ist der Aufwand zur Herstellung dieser Grafiken relativ hoch. Die grafischen Darstellungen werden abgespeichert, damit sie für eine spätere Verwendung wieder verfügbar sind. Am einfachsten ist das, indem die notwendigen Teile des Bildschirminhalts in Dateien gesichert werden. Bei einer notwendigen Neudarstellung genügt dann ein Rückspeichern aus diesen Dateien. Diese Vorgehensweise wird beispielsweise in Turbo-Pascal durch die Prozeduren PUTIMAGE und GETIMAGE sehr gut unterstützt.
Es kann nun vorkommen, daß die gespeicherten Bilder neueren Anforderungen nicht mehr entsprechen. Notwendig könnten größere oder kleinere Formate oder auch Dehnungen werden. Ähnlich sind die Verhältnisse, wenn Grafiken auf Bildschirmen mit verschiedenen Auflösungen dargestellt werden sollen. Das ist beispielsweise der Fall, wenn mit CGA-, EGA- und Hercules-Grafikkarten gearbeitet wurde. Die beste Lösung wäre in diesen Fällen eine Neuanfertigung der Grafiken. Dieser Weg ist aber eventuell auch der aufwendigste. Ist das dazugehörige Programm nicht flexibel genug, muß ein Eingriff in die Programmstruktur vorgenommen werden. Häufig sind auch die Originaldaten, mit deren Hilfe die Grafiken erzeugt wurden, nicht mehr verfügbar.
Es sollen nun Algorithmen vorgestellt werden, die die weitere Bearbeitung von Bildern gestatten. Es ist klar, daß im allgemeinen diese Bearbeitung eine Verschlechterung der Qualität mit sich bringt. Es zeigt sich aber, daß durchaus befriedigende Ergebnisse erreichbar sind, sofern es sich nicht um extreme Skalierungsmaßstäbe handelt.
Bei der Beschreibung der Algorithmen wird auf eine zweifarbige Darstellung orientiert, jedoch sind die Algorithmen auch für mehrfarbige Bilder geeignet. Die Beispielgrafiken wurden mit Hilfe eines Schneider PC 1512 ohne Koprozessor mit CGA-Karte erzeugt. Die benutzte Programmiersprache war Turbo-Pascal, Version 4.0. Die Rechenzeiten beziehen sich auf eine Version der Algorithmen, die alle CGA-Modi des Rechners (einschließlich Farbe) unterstützt, so daß sie etwas höher liegen als bei einem Algorithmus für 2-Farben-Bilder notwendig wäre.

Algorithmus

Die grafische Darstellung auf dem Bildschirm wird als Menge von Rasterpunkten (Pixel) aufgefaßt. Die Bildbearbeitung umfaßt deshalb die Aufgabe, durch eine entsprechende Transformation der Pixelkoordinaten ein neues Bild aufzubauen. Für diese Aufgabe ist es ausreichend, die Transformationen Translation, Skalierung und Rotation zu betrachten. Als Ausgangspunkt der Betrachtungen wird der Bildschirm als Teil eines Koordinatensystems aufgefaßt. Dabei erhält die linke obere Ecke die Koordinate (0,0), die rechte untere Ecke die Koordinate (xmax,ymax). Zur Beschreibung der Transformationsformel wird festgelegt:
– Die linke obere Ecke des zu bearbeitenden Bildes hat die Koordinaten (X1,Y1), die rechte untere Ecke die Koordinaten (X2,Y2).
– Die Translation verschiebt die Koordinate (X1,Y1) auf die Koordinate (Tx,Ty).
– Die Skalierung der x-Achse verändert eine Längeneinheit auf die Länge Sx, die der y-Achse auf die Länge Sy.
– Die Rotation bezieht sich im mathematisch positiven Sinn auf die x-Achse mit dem Scheitelpunkt in der Koordinate (Tx,Ty). Der Rotationswinkel wird mit PHI bezeichnet.
– Mit R wird das physische Höhen/Seiten-Verhältnis des Bildschirms bezeichnet.

Bemerkung: Die Größen Tx, Ty, Sx, Sy und PHI werden als reelle Zahlen aufgefaßt. Bei der rechentechnischen Umsetzung des Algorithmus erfolgt dann stets eine Rundung.
Die transformierten Koordinaten (px,py) des Pixels (x,y) ergeben sich dann aus der Formel (x)
Zu beachten ist, daß die so berechneten transformierten Koordinaten außerhalb des Bildschirms liegen können. Diese Punkte sind dann nicht mehr darstellbar. Die Normierung und die Renormierung des Bildschirms sind notwendig, damit die Längenverhältnisse des Bildes bei Rotation nicht verändert werden (Prinzip: Quadrat bleibt Quadrat).
In einfachster Form läßt sich nun die Verarbeitung des Bildes formulieren als

Algorithmus 1 – Grundalgorithmus:
```
1. Lege Tx, Ty, Sx, Sy, PHI fest.
2. FOR x:=X1 TO X2 DO
     FOR y:=Y1 TO Y2 DO
      BEGIN
        Berechne die transformierten Koordi-
         naten (px,py) nach Formel (x);
        Setze Pixel (px,py);
      END
```

Dieser Algorithmus wird immer dann gute Ergebnisse liefern, wenn sowohl abs(Sx) ≦ 1 als auch abs(Sy) ≦ 1 gilt (Verkleinerung von Bildern). Seine Anwendung auf das Testbild (Bild 1) zeigt Bild 2.

Verbesserungen

Die Qualität des bearbeiteten Bildes kann durch weitere Maßnahmen verbessert werden. Dazu ist es allerdings notwendig, die Struktur der Grafik in die Betrachtungen einzubeziehen. Zu diesem Zweck definieren wir:

Definition:
1. Die Menge der Pixel
BS = { (x,y) : x in [0..xmax], y in [0..ymax] }
heißt Menge zulässiger Pixel.
2. Menge der Pixel
– UK(x,y) = BS ∩ { (x1,y1) : |x1-x| + |y1-y| = 1 }
heißt Kreuzumgebung von (x,y)
– UD(x,y) = BS ∩ { (x1,y1) : |x1-x| = 1 and |y1-y| = 1 }

$$\begin{pmatrix} px \\ py \end{pmatrix} = \underbrace{\begin{pmatrix} Tx \\ Ty \end{pmatrix}}_{\text{Translation}} + \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & R \end{pmatrix}}_{\text{Renormierung des Bildschirms}} \underbrace{\begin{pmatrix} \cos(PHI) & \sin(PHI) \\ \sin(PHI) & -\cos(PHI) \end{pmatrix}}_{\text{Rotation}} \underbrace{\begin{pmatrix} Sx & 0 \\ 0 & Sy \end{pmatrix}}_{\text{Skalierung}} \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1/R \end{pmatrix}}_{\text{Bildschirmnormierung}} \begin{pmatrix} x-x1 \\ y-y1 \end{pmatrix} \qquad (x)$$

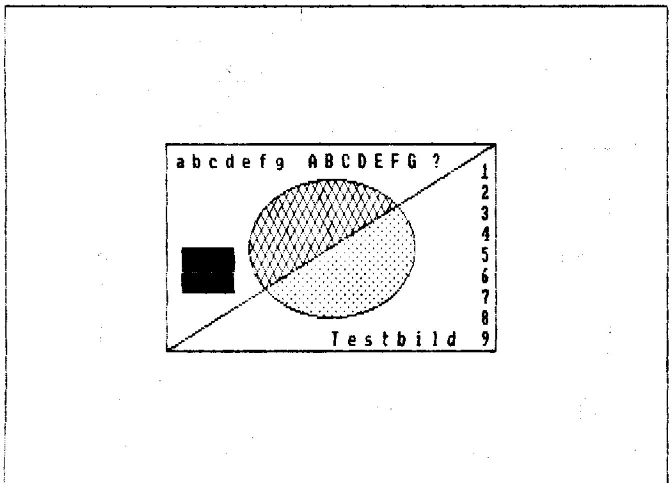


Bild 1 Testbild

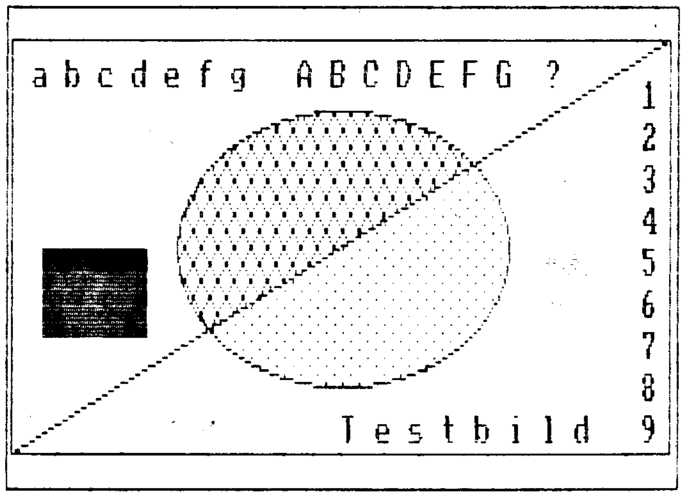


Bild 2 Verdopplung des Testbildes mit Algorithmus 1

heißt Diagonalumgebung von (x,y)
– U(x,y) = UK (x,y) ∩ UD (x,y) heißt Umgebung von (x,y)
3. Ein Pixel (x1,y1) heißt bezüglich (x,y)
– kreuzverbunden, wenn (x1,y1) ∈ UK (x,y)
– diagonalverbunden, wenn (x1,y1) ∈ UD (x,y)
– verbunden, wenn (x1,y1) ∈ U(x,y)
4. Ein Pixel heißt isoliert, wenn U(x,y) = (leere Menge).

Die Bildanalyse stützt sich nun auf die Annahme, daß verbundene Pixel nicht zufällig verbunden, sondern Bestandteil einer Geraden sind. Diese Annahme ist in vielen Fällen plausibel (z. B. Darstellungen von Koordinatensystemen, Balkendiagrammen) oder in anderen Fällen akzeptabel (z. B. bei der Auffassung, daß Schriftzeichen aus Geraden aufgebaut werden können). Der Grundalgorithmus kann deshalb verbessert werden, indem für jedes Pixel seine Umgebung herangezogen wird, die verbundenen Pixel transformiert werden und danach durch Geraden verbunden werden. Je nach der Art der Umgebung können drei weitere Algorithmen formuliert werden.

Algorithmus 2 – Kreuzalgorithmus:

```
1. Lege Tx, Ty, Sx, Sy, PHI fest.
2. FOR x:=X1 TO X2 DO
     FOR y:=Y1 TO Y2 DO
       BEGIN
         Transformiere (x,y) in (px,py) nach
         Formel (x);
         FOR ALL (r ,s) ∈ UK(x,y) DO
           BEGIN
             Transformiere (r ,s) in (pr ,ps)
             nach Formel (x);
             Zeichne eine Gerade von (px,py)
             nach (pr ,ps);
           END
         IF UK(x,y) THEN setze Pixel (px,py);
       END
```

Algorithmus 3 – Diagonalalgorithmus:

```
1. Lege Tx, Ty, Sx, Sy, PHI fest.
2. FOR x:=X1 TO X2 DO
     FOR y:=Y1 TO Y2 DO
       BEGIN
         Transformiere (x,y) in (px,py) nach
         Formel (x);
         FOR ALL (r,s) ∈ UD(x,y) DO
           BEGIN
             Transformiere (r ,s) in (pr ,ps) nach
             Formel (x);
             Zeichne eine Gerade von (px,py)
             nach (pr ,ps);
           END
         IF UD(x,y) THEN setze Pixel (px,py);
       END
```

Algorithmus 4:

```
1. Lege Tx, Ty, Sx, Sy, PHI fest.
2. FOR x:=X1 TO X2 DO
     FOR y:=Y1 TO Y2 DO
       BEGIN
         Transformiere (x,y) in (px,py) nach
         Formel (x);
         FOR ALL (r,s) ∈ U(x,y) DO
           BEGIN
             Transformiere (r,s) in (pr,ps) nach
             Formel (x);
             Zeichne eine Gerade von (px,py)
             nach (pr,ps);
           END
         IF U(x,y) THEN setze Pixel (px,py);
       END
```

Die Algorithmen 2 und 3 sollten nur dann angewendet werden, wenn die Grafiken vorrangig aus senkrechten, waagerechten oder diagonalen Linien bestehen (z. B. Gitternetze). Der Algorithmus 4 liefert im allgemeinen die beste Qualität, benötigt aber auch die längste Zeit. Alle 3 Algorithmen werden dadurch beschleunigt, daß nicht die vollständige Umgebung der Pixel verarbeitet wird, sondern nur diejenigen mit (x,y) verbundenen Pixel, für die die Komponente s ≧ y (vertikale Bearbeitung) bzw. r ≧ x (horizontale Bearbeitung gilt.
Die Anwendung der Algorithmen 2, 3 und 4 zeigen die Bilder 3 bis 5.

Vektorisierung von Bildern

Eine Analyse der Algorithmen 2 bis 4 zeigt, daß eine weitere Beschleunigung der Bildverarbeitung denkbar ist. Es ist generell so, daß die Bearbeitung einer Geraden im Urbild durch eine gestückelte Bearbeitung von jeweils zwei verbundenen Pixeln vorgenom-

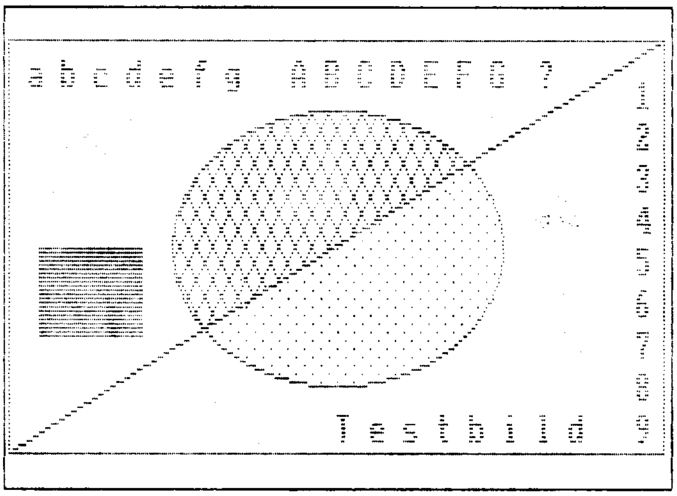


Bild 3 Verdopplung der Größe des Testbildes mit Algorithmus 2

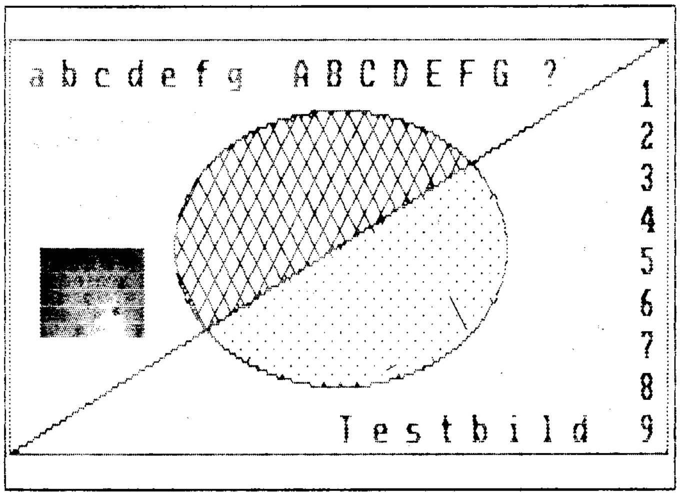


Bild 4 Verdopplung der Größe des Testbildes mit Algorithmus 3

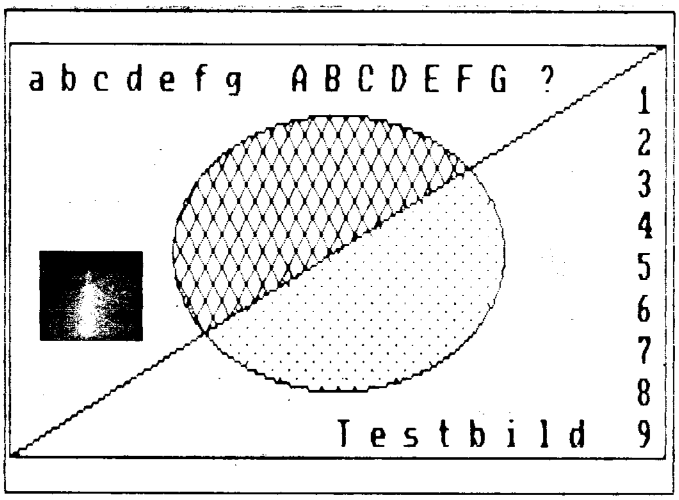


Bild 5 Verdopplung der Größe des Testbildes mit Algorithmus 4

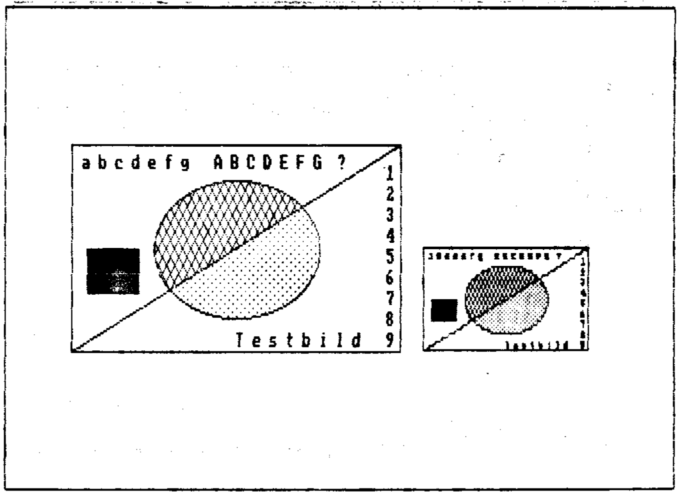


Bild 6 Halbierung der Größe des Testbildes mit Algorithmus 4

men wird. Ausreichend wäre aber die alleinige Bearbeitung des Anfangs- und des Endpunktes dieser Geraden, die dann anschließend wieder durch eine Gerade verbunden werden. Um einen derartigen Algorithmus ausführen zu können, muß eine kompliziertere Bildanalyse vorgenommen werden. Während es für die Algorithmen 1 bis 4 angebracht ist, auf eine pixelweise Speicherung des Bildes zurückzugreifen, (erzeugt zum Beispiel mit der Turbo-Pascal-Prozedur GETIMAGE), wird nun eine Beschreibung in Form zu zeichnender Geraden nötig. Diese Beschreibung ist zur Speicherung eines Bildes genauso gut geeignet wie die pixelweise. Allerdings kann nicht mehr von vornherein angegeben werden, wie groß der dazu notwendige Speicherplatz sein muß. Beispielsweise werden zur pixelweisen Speicherung des Testbildes maximal 16004 Byte benötigt, während der nachfolgend beschriebene Algorithmus 5 eine Datei mit 27148 Byte erzeugt.

Die Bildanalyse erfolgt nach folgendem Schema: Das Bild wird zeilenweise waagerecht, spaltenweise senkrecht sowie diagonal von links und rechts fallend durchmustert. In jeder Suchrichtung wird untersucht, wo eine Gerade beginnt oder endet, und ihre Eckpunkte werden registriert. Geraden, die nur aus einem Pixel bestehen, müssen nur dann registriert werden, wenn dieses Pixel isoliert ist.

Formal wird diese Bildanalyse durch den Algorithmus 5 ausgeführt.

Algorithmus 5 – Bildanalyse:

PROCEDURE analyse(startx,starty,suchrichtung)

1. Für jede Suchrichtung lege ix und iy fest als

$$iy = \begin{cases} 0 & \text{waagerechte Suchrichtung} \\ 1 & \text{sonst} \end{cases}$$

$$ix = \begin{cases} 1 & \text{waagerechte Suchrichtung} \\ 0 & \text{senkrechte Suchrichtung} \\ 1 & \text{von links fallende Suchrichtung} \\ -1 & \text{von rechts fallende Suchrichtung} \end{cases}$$

2. v:=FALSE
3. WHILE startx in [x1..x2] and starty in [y1..y2] DO

```
    BEGIN
      v1:=pixelgesetzt (startx,starty);
      IF v
        THEN BEGIN
          IF not v1
            THEN BEGIN
              Registriere die Gerade (xanf,
                yanf)-(startx-ix,starty-iy);
              xanf:=startx;
              yanf:=starty;
            END
          END
        ELSE IF v1
          THEN BEGIN
            xanf:=startx;
            yanf:=starty;
          END
        startx:=startx+ix;
        starty:=starty+iy;
        v:=v1;
      END
```

4. IF v

```
      THEN registriere die Gerade (xanf,
              yanf)-(startx-ix, starty-iy);
END PROCEDURE
```

Die durch diesen Algorithmus erzeugte Datei dient dann dazu, das gewünschte Bild wieder neu aufzubauen. Der dazugehörige Algorithmus 6 ist denkbar einfach.

Algorithmus 6:

```
WHILE not eof DO
  BEGIN
    Lies die Endkoordinaten (x1,y1),(x2,y2);
    Transformiere sie nach Formel (x) auf die
    Koordinaten
      (px1,py1),(px2,py2);
    Zeichne die Gerade (px1,py1)-(px2,py2);
  END
```

Diskussion der Algorithmen

Problematisch ist die Anwendung der vorgestellten Algorithmen vor allem bei Verkleinerungen des Originalbildes. Sehr schnell kann es dabei zu einer starken Verwischung der Konturen kommen. Ein Beispiel hierfür wird in Bild 6 gegeben.

Es ist auch nicht zu erwarten, daß die Rückwärtsverkleinerung einmal bearbeiteter Bilder wieder das Originalbild ergeben. Ursache dafür ist der Rundungsprozeß bei den verschiedenen Operationen des Bearbeitungsalgorithmus. Wird beispielsweise das Testbild (Bild 1) mit dem Algorithmus 4 auf das Doppelte vergrößert (Bild 5) und anschließend wieder auf die Hälfte verkleinert, ergibt sich die in Bild 7 gezeigte Differenzmenge. Sie ist als Ungenauigkeit des Algorithmus zu interpretieren. Eine noch größere Differenz wird sich ergeben, wenn ein Bild gedreht und anschließend wieder zurückgedreht wird.

Bei extremen Vergrößerungen (siehe Bild 8) dagegen kann die Struktur des Ausgangsbildes verloren gehen. Im allgemeinen erzeugt aber der Algorithmus 4 noch brauchbare Bilder. Das Bild 8 zeigt gleichzeitig, wie die einzelnen Algorithmen vergrößern. Während der Algorithmus 1 keine brauchbare Vergrößerung liefert, ist bei den Algorithmen 2 und 3 das transformierte Bild erkennbar, eine brauchbare Darstellung liefert allerdings erst der Algorithmus 4.

Ein weiteres Problem ergibt sich bei der Rotation des Urbildes. Aufgrund des unterschiedlichen Auflösungsvermögens des Bildschirms in x- und y-Richtung kann es zu einer relativ großen Vergröberung der Bilddarstellung kommen. Ein Beispiel dafür ist das Bild 9, in der das Testbild um 45 Grad gedreht wurde.

Die bisherigen Erfahrungen besagen, daß der Algorithmus 4 und der ihm grafisch gleichwertige Algorithmus 6 die besten Ergebnisse liefern. Bei ihrer Anwendung muß aber folgendes beachtet werden. Der Algorithmus 4 ist immer dann anzuwenden, wenn es um eine einmalige Bearbeitung des Urbildes geht. Algorithmus 6 wird immer dann sinnvoll sein, wenn ein Urbild flexibel mehrfach bearbeitet werden soll. Die Zeiten zum Aufbau eines neuen Bildes betragen nur ungefähr 40 % der Zeit des Algorithmus 4.

Ein Vergleich der Rechenzeiten aller Algorithmen ist in Tafel 1 zu finden. Daraus ist ersichtlich, daß der Algorithmus 6 am schnellsten arbeitet (wenn die Analysezeit nicht beachtet wird). Die gegenüber Algorithmus 4 kürzeren Zeiten der Algorithmen 1 bis 3 sind verbunden mit der schlechteren Qualität der Bilder. Die Rechenzeiten für die Bildverkleinerung sind für die Algorithmen 1 bis 4 gleich, weil hier generell verzichtet wurde, Pixel durch Geraden zu verbinden.

Anzumerken bleibt noch, daß die Bearbeitungszeiten nicht wesentlich von den Bearbeitungsparametern abhängen. Entscheidend ist hier allein die Pixelzahl des Originalbildes. Geringe Differenzen ergeben sich ausschließlich durch die Anwendung des Algorithmus für die Liniendarstellung (Turbo-Pascal-Prozedur LINE), der für verschieden geneigte Linien unterschiedliche Abarbeitungszeiten zu haben scheint.

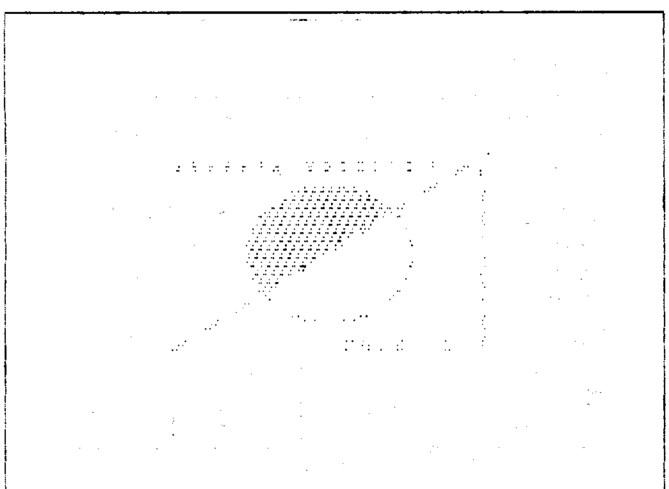


Bild 7 Differenzbild für Algorithmus 4

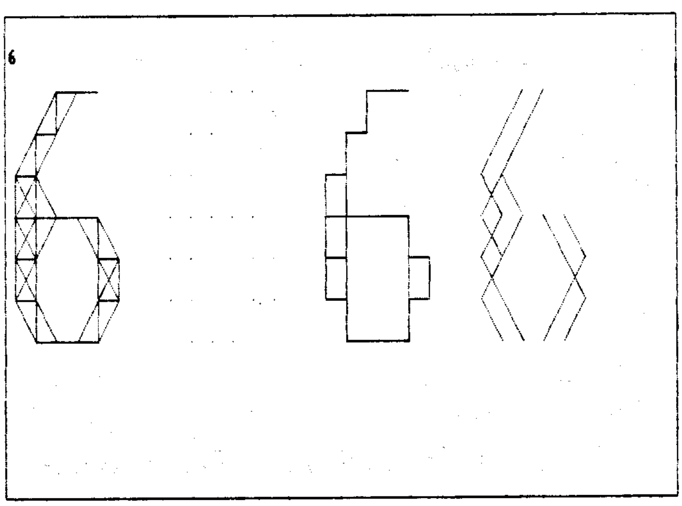


Bild 8 20fache Vergrößerung der Ziffer 6

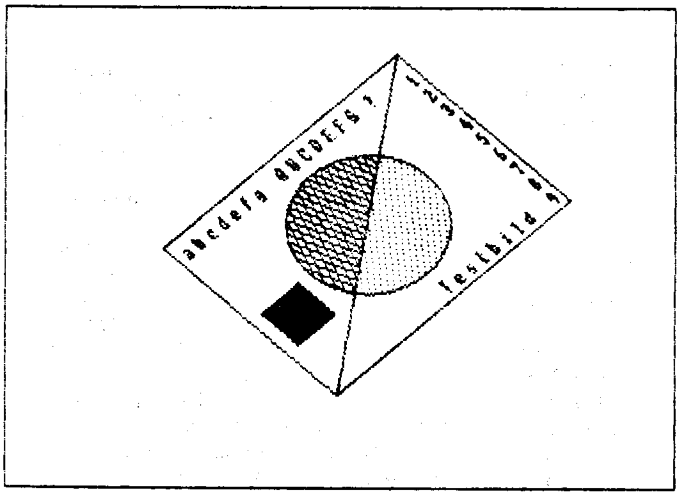


Bild 9 Drehung des Testbildes um 45 Grad

```
PROGRAM demo;

USES crt,graph;

TYPE REAL=SINGLE;             (** falls Co-Prozessor vorhanden **)

PROCEDURE square(x1,y1,laenge:INTEGER);
 VAR x,y : WORD;
 BEGIN
  getaspectratio(x,y);
  rectangle(x1,y1,x1+laenge,y1+round((laenge)*(x+0.0)/y));
  floodfill(x1+1,y1+1,1);
 END;   (* --- square --- *)

PROCEDURE testbild;
 VAR i:INTEGER;
 BEGIN
  rectangle(160,50,478,149);
  line(160,149,478,50);
  circle(320,100,80);
  setfillstyle(8,1);
  floodfill(320,90,1);
  setfillstyle(10,1);
  floodfill(320,110,1);
  outtextxy(170,55,'a b c d e f g   A B C D E F G  ?');
  FOR i:=1 TO 9 DO
   outtextxy(465,50+10*i,chr(i+ord('0')));
  outtextxy(320,140,'T e s t b i l d');
  setfillstyle(1,1);
  square(175,100,50);
 END;   (* --- testbild --- *)


(* --- vereinfachte Form des Algorithmus 4 --- *)

PROCEDURE zeige(tx,ty,sx,sy,phi:REAL; pbild:POINTER);
 TYPE barr = ARRAY[1..1] of BYTE;
      pbarr = ^barr;
 VAR argument,xyrate,xyrate1,
     p1x,p2x,p1y,p2y,pxc,pyc,
     sxc,sys,syc,sxs           : REAL;
     hoehe,breite,bbreite      : LONGINT;
     p                         : pbarr;
     xmax,ymax,xasp,yasp       : WORD;
     i,j                       : INTEGER;
     null                      : BYTE;
     kleiner                   : BOOLEAN;

     PROCEDURE newpoint(x,y:INTEGER; VAR px,py:REAL);
     (* Koordinaten des transformierten Pixels *)
     VAR xx,yy : REAL;
     BEGIN
      xx:=x;
      yy:=y*xyrate1;
      px:=pxc+xx*sxc+yy*sys;
      py:=pyc+yy*syc-xx*sxs;
     END;   (* newpoint *)

FUNCTION pixelgesetzt(rx,ry:INTEGER):BOOLEAN;
(* Test, ob Pixel zulaessig und gesetzt *)
VAR index:WORD;
BEGIN
 index:=5+ry*bbreite+(rx shr 3);
 pixelgesetzt:=((p^[index] shr ((7-(rx mod 8))) and 1)<>0)
                and (rx<=breite) and (rx>=0)
                and (ry<=hoehe) and (ry>=0)
END;   (* pixelgesetzt *)

PROCEDURE rahmen;
(* Loeschen des neuen Bildbereichs *)
VAR pppp  : ARRAY[1..5] of pointtype;
    xx,yy : REAL;
    fills : fillsettingstype;
BEGIN
 newpoint(0,0,xx,yy);
 pppp[1].x:=round(xx); pppp[1].y:=round(yy);
 newpoint(breite,0,xx,yy);
 pppp[2].x:=round(xx); pppp[2].y:=round(yy);
 newpoint(breite,hoehe,xx,yy);
 pppp[3].x:=round(xx); pppp[3].y:=round(yy);
 newpoint(0,hoehe,xx,yy);
 pppp[4].x:=round(xx); pppp[4].y:=round(yy);
 pppp[5]:=pppp[1];
 getfillsettings(fills);
 setcolor(0);
 setfillstyle(1,0);
 fillpoly(5,pppp);
 setcolor(1);
 setfillstyle(fills.pattern,fills.color);
END;   (* rahmen *)

 BEGIN
  null:=0;
  xmax:=getmaxx;
  ymax:=getmaxy;
  getaspectratio(xasp,yasp);
  xyrate  :=(xasp+0.0)/yasp;
  xyrate1 :=1.0/xyrate;
  argument:=pi*phi/180.0;
  sxc:=sx*cos(argument);          sys:=sy*sin(argument);
  syc:=sy*cos(argument)*xyrate; sxs:=sx*sin(argument)*xyrate;
  pxc:=tx; pyc:=ty;
  p:=pbarr(pbild);
  IF ( (sx=1.0) AND (sy=1.0) AND (phi=0.0) )
   THEN BEGIN
         (* Bild muss nur verschoben werden *)
         putimage(round(tx),round(ty),p^,normalput);
         EXIT;
        END;
  kleiner:=not ((sx>0.0) and (sx<1.0) and (sy>0.0) and (sy<1.0));
                  (* kleiner=TRUE, falls keine Verkleinerung vorliegt *)
  hoehe:=0;
  breite:=0;
  move(p^[1],breite,2);              (* Bestimmung des Bildformats *)
  move(p^[null+3],hoehe,2);
  bbreite:=(breite div 8)+1;
  rahmen;

 FOR i:=0 TO hoehe DO
  FOR j:=0 TO breite DO
   IF pixelgesetzt(j,i)
    THEN BEGIN
          setcolor(1);
          newpoint(j,i,p1x,p1y);
          line(round(p1x),round(p1y),round(p1x).round(p1y));
          IF kleiner
           THEN BEGIN
                 IF pixelgesetzt(j+1,i+1)
                  THEN BEGIN
                        newpoint(j+1,i+1,p2x,p2y);
                        line(round(p1x),round(p1y),round(p2x),round(p2y
                       END;
                 IF pixelgesetzt(j+1,i)
                  THEN BEGIN
                        newpoint(j+1,i,p2x,p2y);
                        line(round(p1x),round(p1y),round(p2x),round(p2y
                       END;
                 IF pixelgesetzt(j-1,i+1)
                  THEN BEGIN
                        newpoint(j-1,i+1,p2x,p2y);
                        line(round(p1x),round(p1y),round(p2x),round(p2y
                       END;
                 IF pixelgesetzt(j,i+1)
                  THEN BEGIN
                        newpoint(j,i+1,p2x,p2y);
                        line(round(p1x),round(p1y),round(p2x),round(p2y
                       END;
                END;
         END;
END;   (* --- zeige --- *)

(* ------------------------------------------ *)


VAR tx,ty,sx,sy,phi  : REAL;
    pbild            : POINTER;
    grdriver,grmode  : INTEGER;
    l                : WORD;
BEGIN          (* Beginn des Demonstrationsprogramm *)
 clrscr;
 write('Verschiebung auf x = '); readln(tx);
 write('                 y = '); readln(ty);
 write('Streckungsfaktor x = '); readln(sx);
 write('Streckungsfaktor y = '); readln(sy);
 write('Drehungswinkel in Grad = '); readln(phi);
 grdriver:=cga;
 grmode  :=cgahi;
 initgraph(grdriver,grmode,'c:\turbo\tpu');
 clearviewport;
 testbild;
 l:=imagesize(160,50,478,149);
 getmem(pbild,l);
 getimage(160,50,478,149,pbild^);
 zeige(tx,ty,sx,sy,phi,pbild);
 readln;
 freemem(pbild,l);
END.
```

Bild 10 Demo-Programm für den Algorithmus 4

Tafel 1

Algorithmus	Maßstab Sx = Sy	PHI Grad	Zeit s	Bemerkung
1	0,5 2,0 1,0	0 0 45	64 64 68	Bild 2
2	0,5 2,0 1,0	0 0 45	64 110 119	Bild 3
3	0,5 2,0 1,0	0 0 45	64 103 111	Bild 4
4	0,5 2,0 1,0	0 0 45	64 149 161	Bild 5 Bild 9
5	–	–	163	Analysezeit Algorithmus 5
6	0,5 2,0 1,0	0 0 45	45 47 53	entspricht Bild 5

Fortsetzung auf S. 26

Einführung in MS-DOS

Teil 3

Wolfram Schulze, Uwe Schulze, Berlin

Der dritte Teil des Kurses widmet sich dem Prompt-Befehl, zeigt, was es mit dem ANSI-Treiber auf sich hat und stellt die kleine, aber leistungsfähige Kommandosprache von DOS vor.

Nutzung des Prompt-Befehls

Als Prompt wird das Bereitschaftszeichen von DOS bezeichnet. Es zeigt das aktuelle Laufwerk, meistens den aktuellen Pfad und ein Größerzeichen. Dem Nutzer ist es überlassen, den Prompt nach eigenem Belieben zu verändern. Versuchen Sie es doch einmal mit

prompt alles roger

und drücken Sie ein paarmal <ENTER>, um zu sehen, was passiert. Ganz lustig ist das ja, aber nicht allzu hilfreich, da wir nicht mehr wissen, auf welchem Laufwerk wir uns befinden. Mit

prompt

ohne Parameter läßt sich wieder die Standardanzeige einstellen. Außer beliebigem Text kann der Prompt aber noch eine Reihe von Systeminformationen anzeigen. Dafür sind einige Sonderzeichen reserviert. Sie bestehen jeweils aus einem Dollarzeichen ($) und einem Buchstaben. Im folgenden sind die möglichen Kombinationen aufgelistet:

$b das Zeichen |
$d das Datum
$e das Zeichen Escape (1BH)
$h letztes Zeichen löschen
$g das Zeichen >
$l das Zeichen <
$n das aktuelle Laufwerk
$p das aktuelle Verzeichnis
$q das Zeichen =
$s das Leerzeichen
$t die Uhrzeit
$v die DOS-Version
$_ neue Zeile (CR/LF)
$$ das Zeichen $

Der Standardprompt, den man erhält, wenn man PROMPT ohne Parameter aufruft, zeigt das Laufwerk und das Größerzeichen. Er entspricht damit der Einstellung prompt ng. Allgemein üblich ist es aber, den aktuellen Pfad anzuzeigen. Dazu dient die Anweisung

prompt pg

wie sie auch in unserer Datei AUTOEXEC-.BAT eingetragen ist. Auf diese Art und Weise lassen sich beliebig komfortable Prompts einrichten, etwa

prompt Zeit:t_Datum$d$_pg

Vergessen Sie über das Experimentieren aber nicht, daß der Prompt nicht nur nett aussehen, sondern auch zweckmäßig sein soll.

Der Prompt-Befehl und ANSI.SYS

Die Bildschirmsteuerung unter MS-DOS ist nicht besonders komfortabel. Außer dem Piepton (Bell), dem Wagenrücklauf (Carriage Return/Linefeed) und dem Löschen des letzten Zeichens (Backspace) werden keine Steuerzeichen interpretiert. Deshalb kann ein zusätzliches Gerätesteuerprogramm (Treiber, englisch Device Driver) verwendet werden, das weitere Steuerbefehle verarbeitet. Man hält sich damit an einen Normvorschlag für Terminal-Steuerbefehle (Normvorschrift X 3.64: Additional Controls for Use with ASCII) der amerikanischen Standardisierungsbehörde ANSI (American National Standards Institute). Deshalb wird der Treiber auch kurz ANSI-Treiber genannt und wir finden ihn auf der Systemdiskette unter dem Namen ANSI.SYS. Um seine Dienste in Anspruch zu nehmen, muß er mit

device=ansi.sys

in die Konfigurationsdatei CONFIG.SYS (siehe Teil 2) eingetragen werden. Steht der ANSI-Treiber nicht im Hauptverzeichnis, so kann auch sein Pfad angegeben werden, beispielsweise

device=\dos\ansi.sys

Anschließend muß das System neu gestartet werden (<CTRL> <ALT> <DEL>, <Strg> <ALT> <Entf> oder Reset-Taste); nur so sind neue Treiber zu installieren. Ab sofort untersucht DOS alle zum Bildschirm ausgegebenen Zeichen auf Steuerfolgen. Da fast alle Tasten in ihrer eigentlichen Bedeutung (Buchstaben, Zahlen, Sonderzeichen) gebraucht werden, stehen nicht genug Steuerzeichen zur Verfügung. Deshalb werden Folgen von Zeichen verwendet, die stets durch das Zeichen Escape (ASCII-Code 1B hex. = 27 dezimal) und die öffnende eckige Klammer [(ASCII-Code 5B hex. = 91 dezimal) eingeleitet werden. Daran erkennt der ANSI-Treiber, daß die folgenden Zeichen nicht als Zahlen, Buchstaben etc., sondern als Steuerfolge zu interpretieren sind. Das einleitende Escape steht auch für die Bezeichnung Escape-Sequenzen (Sequenz (engl.) = Folge, Ordnung). Tafel 1 gibt einen Überblick über die möglichen Befehlsfolgen. Sie müssen nur noch auf den Bildschirm ausgegeben werden und bewirken das entsprechende Kommando. In einer Programmiersprache können die normalen Ausgabeanweisungen verwendet werden, also *write* in Pascal oder *printf* in C. (später ein Beispiel dazu).

Aber auch im Betriebssystem gibt es Kommandos, die Ausgaben auf den Bildschirm bewirken. Erinnern Sie sich bitte an die letzte Kurs-Folge. Mit

echo Dies ist eine wunderschöne Ausgabe

erhalten Sie eben diese Ausgabe auf den Bildschirm. Leider läßt sich das Escape nicht darstellen; das Betätigen der Escape-Taste (<ESC>) bringt nicht den erhofften Effekt – DOS nutzt die Taste zum Abbrechen von Befehlen. Der Echo-Befehl scheidet also aus. Mit TYPE lassen sich Dateien auf den Bildschirm ausgeben, aber diese müssen erst mit einem Texteditor erstellt werden, und dann

Tafel 1 Überblick über Befehlsfolgen für ANSI.SYS

```
ESC [ x; y H      Setzen des Cursors an Zeile x, Spalte y
ESC [ x; y f      gleiche Wirkung
ESC [ x A         Bewegen des Cursors um x zeilen nach oben
ESC [ x B         Bewegen des Cursors um x zeilen nach unten
ESC [ x C         Bewegen des Cursors um x Spalten vor
ESC [ x D         Bewegen des Cursors um x Spalten zurück
ESC [ 6 n         Aktuelle Cursorposirion abfragen (nicht mit
                  Prompt verwenden)
ESC [ s           Sichern der aktuellen Cursorposition
ESC [ x; y R      Anzeigen der aktuellen Cursorposition
ESC [ u           Setzen des Cursors an die Stelle des
                  letzten Sicherns
ESC [ 2 J         Bildschirm löschen, Cursor home
ESC [ k           Bildschirm ab Cursorposition löschen
ESC [ = x h       Bildschirmmodus setzen
                  x     Wirkung
                  0     40x25 Zeichen schwarz/weiß
                  1     40x25 Zeichen Farbe
                  2     80x25 Zeichen schwarz/weiß
                  3     80x25 Zeichen Farbe
                  4     Grafik: 320x200 Punkte Farbe
                  5     Grafik: 329x200 Punkte schwarz/weiß
                  6     Grafik: 640x200 Punkte schwarz/weiß
ESC [ = x l       Bildschirmmodus zurücksetzen
ESC [ x; ...x m   Zeichenattribute setzen
                  x     Wirkung
                  0     zurücksetzen, weiß auf schwarz
                  1     höhere Helligkeitsstufe
                  2     niedrigere Helligkeitsstufe
                  3     Kursivschrift
                  4     unterstrichen (nur monochrom)
                  5     blinkend
                  6     schnelles Blinken
                  7     invers
                  8     löschen
                  48    hochgestelltes Zeichen
                  49    tiefgestelltes Zeichen
                  Vordergrund     Hintergrund
                  30              40              schwarz
                  31              41              rot
                  32              42              grün
                  33              43              gelb
                  34              44              blau
                  35              45              violett
                  36              46              kobaltblau
                  37              47              weiß
ESC [ x; ... p    Tastatur neu belegen
                  x gibt den Tastencode an, es folgt die
                  neue Belegung
```

gibt es die gleichen Probleme mit nichtdruckbaren Zeichen, wie Escape. Die Überschrift verrät es: Der Prompt-Befehl kann hier Abhilfe leisten. Seine eigentliche Aufgabe besteht bekanntlich in der Einstellung der DOS-Bereitschaftsanzeige; der Nebeneffekt der Ausgabe auf den Bildschirm wird für die Escape-Sequenzen genutzt. Vorteilhaft vor allem, daß das Escape-Zeichen eine eigene Darstellung erhält, nämlich $e (Dollarzeichen und kleines e). Wie Sie Tafel 1 entnehmen können, wird mit ESC [2 J der Bildschirm gelöscht. Die Leerzeichen werden nur zur besseren Lesbarkeit eingefügt und dürfen sich nicht im Kommando befinden, das folgendermaßen aussieht (die eckige Klammer erreichen Sie, wie in Teil 1 beschrieben, mit <ALT> 91):

prompt$e[2J

Wenn Sie alles richtig gemacht haben, ist der Bildschirm jetzt gelöscht und der Cursor steht links oben in der Ecke. Auffallen wird, daß kein Prompt mehr zu sehen ist – ihn haben wir mit dieser Anweisung außer Kraft gesetzt. Um Laufwerk und Pfad wieder angezeigt zu bekommen, muß

prompt$p$g

eingegeben werden. Vorläufig reicht aber auch einfach **prompt**, denn auf den Pfad können wir bei den folgenden Übungen verzichten. Hat der Befehl nicht die erhoffte Aktion ausgeführt, so überprüfen Sie bitte folgende Fehlerquellen: Steht der Treiber ANSI.SYS in der Konfigurationsdatei CONFIG.SYS (wie oben beschrieben)? Und haben sie weitere Programme geladen, die ihrerseits die Bildschirmausgabe kontrollieren?

Zumeist ist es der Norton-Commander (den wir uns in einer weiteren Kurs-Folge noch genauer ansehen werden), es können aber auch andere residente Programme wie Side-Kick oder Superkey daran Schuld sein. Entfernen Sie also zunächst einmal den Norton-Commander (Funktionstaste F10) und probieren Sie es noch einmal.

Nun ist das Löschen des Bildschirms keine allzu hilfreiche Anwendung – dafür gibt es ja den CLS-Befehl. Deshalb wollen wir als nächstes einmal einen anderen Bildschirmmodus einstellen, nämlich mit nur 40 Zeichen auf einer Zeile, die man dann auch ohne Brille ganz gut lesen kann. Dazu haben Sie

prompt $e[=0h

zu schreiben. Auch hier muß der Prompt wieder hergestellt werden, indem prompt eingegeben wird. Inzwischen müßten Sie in der Lage sein, aus Tafel 1 festzustellen, wie in den normalen Bildschirmmodus zurückgeschaltet werden kann. Normal sind 25 Zeilen zu 80 Zeichen (80 × 25). Da es in DOS häufig mehrere Wege zum Ziel gibt, können Sie das gleiche aber auch mit

mode 80

erreichen – in einer späteren Folge wird gezeigt, was es damit auf sich hat. Jetzt dürfte es keine Schwierigkeiten mehr bereiten, beliebige andere Einstellungen vorzunehmen, beispielsweise

prompt $e[7m

zur Einstellung einer inversen Darstellung. Ihnen wird auffallen, daß nicht sofort der ganze Bildschirm invers ist, sondern nur die Ausgaben ab aktueller Cursorposition.

Weitere Möglichkeiten ergeben sich, wenn Sie über einen Farbbildschirm verfügen. Das Auge freut sich sicherlich über gelbe Schrift auf blauem Grund, die mit

prompt $e[33;44m

eingestellt wird. Der Befehl kann gefahrlos auch auf einen Schwarzweißbildschirm angewendet werden. Er erzeugt Graustufen und sieht – je nach Bildschirmkarte – ganz passabel aus. Eine Einstellung ganz besonderer Art wird von

prompt $e[31;41m

vorgenommen. Probieren Sie es doch einmal aus. Die anschließende Eingabe müssen Sie blind vornehmen oder die Reset-Taste betätigen. Warum? Rote Schrift auf rotem Grund ...

ANSI.SYS ist aber nicht nur für den Bildschirm verantwortlich, sondern auch für die Tastatur; Bildschirm und Tastatur werden zusammen Konsole genannt, und ANSI.SYS ist ein Konsol-Treiber.

Was läßt sich an einer Tastatur ändern? Nun, die Belegung der Tasten. Warum? Erinnern sie sich der Suche des Backslashs (\), der auf vielen Tastaturen nur schwer zu erreichen ist. Oder aber denken Sie an die Möglichkeit, eine ganze Zeichenkette (etwa einen Programmnamen) auf nur eine Taste zu legen. Dazu müssen Sie den Code wissen, den diese Taste abgibt. Alle Tasten, die ein druckbares Zeichen darstellen, können Sie unter ihrem ASCII-Code erreichen. Sie benötigen dazu eine ASCII-Tabelle, die in kaum einer Dokumentation fehlt. Dort werden Sie das kleine **u** als eine dezimale 117 und das kleine **x** als 120 finden. Als erstes fällt einem sicherlich ein Beispiel aus dem Leben ein; um also ein **x** für ein **u** zu erhalten, brauchen Sie nur noch

prompt $e[117;120p

einzugeben (siehe auch Tafel 1). Wie Sie sich selbst überzeugen, können trifft die Einstellung jetzt nur für das kleine u zu, für den Großbuchstaben ist die gleiche Prozedur mit seinem Tastencode nötig. Wenn Sie dieses Beispiel nicht überzeugt, so möchte ich Ihnen auch eine sinnvolle Anwendung vorstellen. Wir wenden uns deshalb dem Backslash zu und legen ihn auf das Ausrufezeichen, das uns damit verlorengeht:

prompt $e[33;92p

Beachten Sie bitte, daß vor dem letzten Buchstaben (in diesem Falle das **p** zur Kennzeichnung der Umbelegung) kein Semikolon steht. Da kaum ein Zeichen entbehrlich ist, wäre es viel interessanter, die Funktionstasten zu belegen. Diese Steuertasten besitzen einen sogenannten erweiterten Tastaturcode. Er beginnt stets mit einer Null und es folgt ein weiteres Zeichen (SCAN-Code). Gleiches gilt auch für die Buchstaben in Verbindung mit der ALT- oder Control-(Strg-)Taste. Für eine vollständige Tabelle fehlt uns der Platz – Ihnen bleibt nur der Blick in die Systemliteratur. Trotzdem als kleine Hilfe die wichtigsten Belegungen:

<F1> bis <F10>	Code: 0;59 bis 0;68
<F1> bis <F10> mit <SHIFT>	Code: 0;84 bis 0;93
<F1> bis <F10> mit <CONTROL>	Code: 0;94 bis 0;103
<F1> bis <F10> mit <ALT>	Code: 0;104 bis 0;113
<ALT> <Q>	Code 0;16
<ALT> <W>	Code 0;17
.	
<ALT> <D>	Code 0;32
<ALT> <F>	Code 0;33

Sie sehen, daß die Codes nicht aufsteigend nach dem Alphabet, sondern nach der Anordnung auf der Tastatur vergeben worden sind. Die Abtrennung mit Semikolon wurde gewählt, da sie auch in den Escape-Sequenzen so gefordert ist. Damit Sie einen Vergleich haben, hier der Befehl um den Backslash auf die F1-Taste zu legen:

prompt $e[0;59;92p

Damit verliert die F1-Taste natürlich die in Teil 1 beschriebene Möglichkeit, einzelne Zeichen aus dem Tastaturpuffer zu holen. Und natürlich wird auch deutlich, daß sich solch eine Einstellung nicht mit dem Norton-Commander verträgt, der seine eigene Belegung der F1-Taste durchsetzen würde. Etwas sinnvoller lassen sich die Funktionstasten mit einem ganzen Kommando belegen, zum Beispiel mit dem DIR-Kommando, wiederum auf die Taste F1:

prompt $e[0;59;"dir";13p

Die dezimale 13 hinter dem Kommando ist ein Wagenrücklauf (Carriage Return) und bewirkt, wie das Betätigen der Enter-Taste, die sofortige Ausführung des Kommandos. Genausogut ist es auch möglich, auf die Eingabe weiterer Parameter von Hand zu warten. Das Kommando

prompt $e[0;60;"type"p

legt auf F2 den Befehl TYPE und wartet beim Betätigen der Taste auf Eingabe eines Dateinamens. Das Belegen der Tasten mit Kommandos beschränkt sich nicht auf die Funktionstasten. Dazu nutzt man die Mehrfachbelegung der Tasten, also <CONTROL> <Buchstabe> oder <ALT> <Buchstabe>. So stehen ausreichend Tasten für eigene Ideen zur Verfügung. Versuchen Sie es mal mit dem Kommando FORMAT A: auf die Tasten <ALT> <F>:

prompt $e[0;33;"format a:";13p

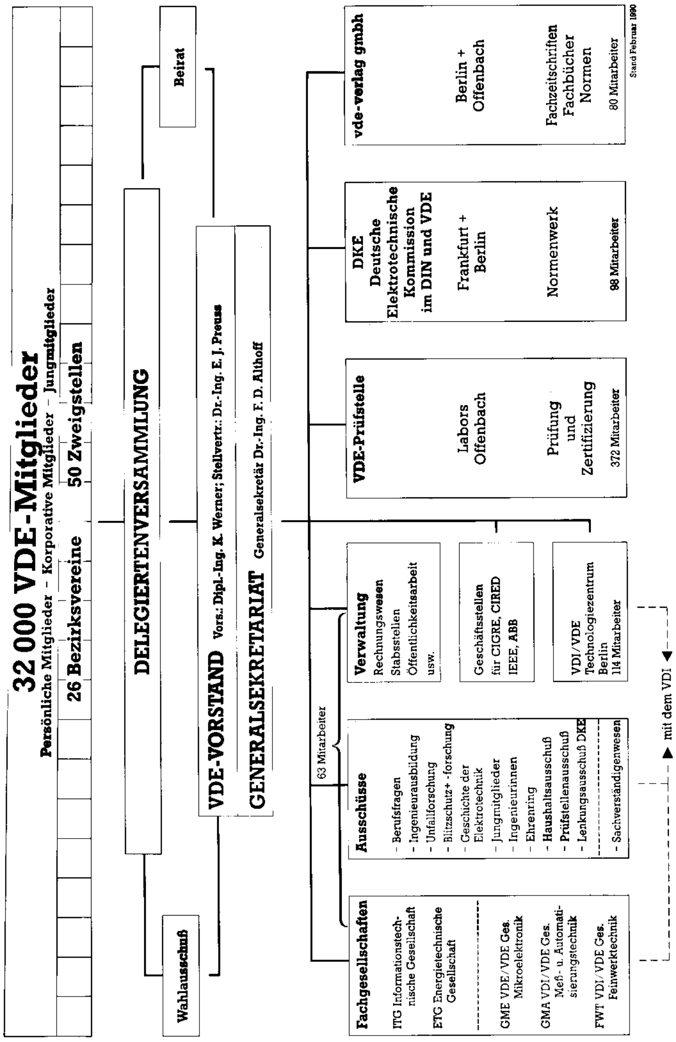
oder mit dem Aufruf von dBase auf <ALT> <D> mit

prompt $e[0;32;"dbase";13p

Aufmerksame Leser werden sicher auf die Idee kommen, nicht nur Systemkommandos, sondern auch sonst benötigte Zeichenketten fest auf Tasten zu legen, zum Beispiel „BEGIN" für Pascal-Programmierer, „LIST" für dBase oder „Mit freundlichen Grüßen" für eine beliebige Textverarbeitung. Hier werden Sie enttäuscht. Der ANSI-Treiber tut seine Arbeit nur auf der Ebene des Betriebssystems, innerhalb von Programmen stehen Ihnen die Belegungen nicht zur Verfügung, da die meisten Programme eigene Ein-/Ausgaberoutinen benutzen (Ausnahmen bestätigen wie immer nur die Regel).

Es gibt allerdings Hilfsprogramme, die sogenannte Tastaturmacros, also die Belegung von Tasten mit Zeichenketten in sehr komfortabler Form gestatten. Dazu gehören Super-Key und Keyworks. Und noch ein Hinweis: Es steht nicht unbegrenzt Platz für die Umbelegung von Tasten zur Verfügung. In DOS 3.x werden Sie im Schnitt auf 30, höchstens 64 Redefinitionen kommen. Aber das sollte ja ausreichend sein. Ist der dafür vorgesehene Puffer voll, werden die danach folgenden Angaben ohne Fehlermeldung ignoriert.

Haben Sie ein paar sinnvolle Belegungen gefunden, so schreiben Sie diese am besten gleich in die AUTOEXEC.BAT, um sie nach jedem Systemstart wieder zur Verfügung zu



Wir bringen Sie mit den besten Fachleuten zusammen

Für Mitgliederfragen steht Ihnen zur Verfügung:
Ursula Gluske-Tibud.

Auch ohne weiteres Wissen können wir schon Batch-Dateien schreiben. Und eine ganz besondere haben wir schon kennengelernt: AUTOEXEC.BAT. Als nächstes soll eine Batch-Datei angelegt werden, die uns die verfügbaren externen Befehle auf der Festplatte anzeigt. Dazu brauchen wir nun aber doch einen Texteditor – der COPY-Befehl ist für größere Arbeiten nicht geeignet. Schreiben Sie also mit dem Editor Ihrer Wahl (Wordstar, Norton-Editor, Turbo-Editor) die neue Datei EXDIR.BAT mit dem Inhalt:

dir \dos*.com /w

Anschließend können Sie die Datei mit ihrem Namen – also EXDIR – aufrufen, und es werden alle COM-Dateien aus dem Verzeichnis DOS (und dort stehen gerade die externen DOS-Befehle) in Kurzform angezeigt. Bei nur einem Befehl in der Kommandodatei ist die Arbeitserleichterung nicht sonderlich groß; ist. Als nächstes soll verhindert werden, daß man auch die CONFIG.SYS damit ansehen kann. Alle anderen Dateien ja, aber die Konfigurationsdatei soll nicht zugänglich sein. Dazu sollte es genügen, das Kommando

if %1 == config.sys echo Diese Datei ist tabu

voranzustellen (beachten Sie bitte, daß es sich um zwei Gleichheitszeichen handelt). Beim Aufruf von **test config.sys** werden Sie feststellen, daß die Ausschrift wirklich kommt, die Datei aber anschließend trotzdem gezeigt wird (sofern sie vorhanden ist). Ursache dafür ist, daß alle Befehle nacheinander durchlaufen werden. Um dem zu begegnen, kann mit **GOTO** verzweigt werden, indem eine Sprungmarke angegeben wird. Sie kann beliebig lang sein, aber nur 8 Zeichen werden zur Auswertung herangezogen. Die Marke selbst wird von einem Doppelpunkt eingeleitet. Wird sie nicht gefunden, so bricht det. Zur besseren Lesbarkeit ist aber auch if "%1" == "" ... möglich. Der Batch zum Anzeigen von 1 bis 3 Dateien hat dann folgendes Aussehen:

```
if exist %1 type %1
if "%2" == "" goto ende
if exist %2 type %2
if "%3" == "" goto ende
if exist %3 type %3
:ende
```

Wenn Sie genau aufgepaßt haben, werden Sie feststellen, daß jetzt doppelt geprüft wird. Denn ein nicht übergebener Parameter wird als leere Zeichenkette interpretiert und deshalb als Datei nicht gefunden, was auch keine Fehlermeldung auslöst. In vielen Fällen ist es aber wichtig, auf die Existenz eines Parameters zu prüfen. So würde die Anweisung **format %1:** beim Fehlen des Laufwerksparameters **format :**ergeben. Wird die Existenz des Para-

haben, denn selbstverständlich gehen sie verloren, wenn der Rechner abgeschaltet wird.
Zu guter Letzt sei noch einmal daran erinnert, daß die Ausgabe von Steuersequenzen mit der eigentlichen Aufgabe des Prompt-Befehls nichts zu tun hat; er wird sozusagen zweckentfremdet benutzt. Der Vorteil liegt darin, daß wir nicht extra eine Programmiersprache bemühen müssen, sondern auf DOS-Kommandoebene arbeiten können. In einer Programmiersprache können die Steuerzeichen einfach mit dem Ausgabebefehl auf den Bildschirm ausgegeben werden. Die einzige Schwierigkeit besteht darin, daß der Programmierer wissen muß, wie die nichtdruckbaren Zeichen (vor allem ESCAPE) ausgegeben werden. Die Belegung der F1-Taste mit dem DIR-Befehl sei hier als Beispiel in Pascal und C gezeigt:

```
write(chr(27),'[0;59;"dir";13p');          {Pascal}
printf("%c[0;59;\"dir\";13p",27);  /*C*/
```

Achten Sie aber darauf, daß der Compiler die Programmierschnittstelle von DOS (BDOS) verwendet und nicht (an ANSI.SYS vorbei) auf das ROM-BIOS zugreift. Für Turbo-Pascal heißt das, auf die Unit CRT zu verzichten.

Stapelverarbeitung

Der Ihnen vielleicht etwas seltsam anmutende Begriff Stapelverarbeitung stammt noch aus Zeiten, als man wahrhaftig Lochkartenstapel in seinem Rechenzentrum abgab, um dafür (Tage später) eine Druckliste zu erhalten. Damit sich das aber nicht allzu profan anhört, sagte man auch Job dazu. Die PC-Nutzer geben sich modern und sagen Batch, was auch nichts anderes heißt als Stapel oder Stoß (Papier). Darunter wird eine Folge von Kommandos auf DOS-Ebene verstanden, die ohne Zutun des Nutzers als Block abgearbeitet werden. Ziel ist es, stets wiederkehrende Arbeiten zu automatisieren. Dazu können alle bisher besprochenen Befehle (interne und externe) sowie beliebige Programme verwendet werden. Zusätzlich stellt DOS noch Steuerstrukturen zur Verfügung, die die eigentliche Kommandosprache ausmachen, und um die soll es in diesem Abschnitt gehen. Vom Umfang her ist die DOS-Kommandosprache recht klein (verglichen mit denen anderer Betriebssysteme), aber sie kann sehr mannigfaltig eingesetzt werden und bildet so ein mächtiges Werkzeug.
Auch ohne weiteres Wissen können wir schon Batch-Dateien schreiben. Und eine ganz besondere haben wir schon kennengelernt: AUTOEXEC.BAT. Als nächstes soll eine Batch-Datei angelegt werden, die uns die verfügbaren externen Befehle auf der Festplatte anzeigt. Dazu brauchen wir nun aber doch einen Texteditor – der COPY-Befehl ist für größere Arbeiten nicht geeignet. Schreiben Sie also mit dem Editor Ihrer Wahl (Wordstar, Norton-Editor, Turbo-Editor) die neue Datei EXDIR.BAT mit dem Inhalt:

```
dir \dos\*.com /w
```

Anschließend können Sie die Datei mit ihrem Namen – also EXDIR – aufrufen, und es werden alle COM-Dateien aus dem Verzeichnis DOS (und dort stehen gerade die externen DOS-Befehle) in Kurzform angezeigt. Bei nur einem Befehl in der Kommandodatei ist die Arbeitserleichterung nicht sonderlich groß; wie weitere Befehle verwendet werden können, war schon bei der AUTOEXEC zu sehen. Was jedoch fehlt, sind Steuerstrukturen für die Wiederholung von Kommandos und zum Prüfen von Bedingungen. Für letzteres gibt es in DOS den **IF**-Ausdruck. Es können drei verschiedene Arten von Bedingungen getestet werden: die Existenz von Dateien (mit EXIST), die Rückgabe von Fehlernummern durch Programme (ERRORLEVEL) und die Gleichheit von Zeichenketten (==), was besonders für Parameter und Dateinamen verwendet werden kann. Durch den Einsatz von **NOT** können die Aussagen auch jeweils negiert werden. Wenn Sie die Datei CONFIG.SYS listen wollen, aber keine Fehlerausschrift, wenn diese nicht vorhanden ist, dann geht das mit

```
if exist config.sys type config.sys
```

Diese Befehlsfolge kann auch interaktiv eingegeben werden. Im folgenden wird nicht weiter darauf hingewiesen, diese Anweisungen in eine BAT-Datei zu schreiben und zu starten.
Genauso kann auch eine Aktion ausgeführt werden, wenn die Datei nicht vorhanden ist. Oder gleich in beiden Fällen:

```
if exist config.sys type config.sys
if not exist config.sys
echo CONFIG.SYS ist nicht vorhanden
```

Die Unterdrückung der Anzeige der Befehle erfolgt mit **echo off**. Dieser Befehl steht in der Regel in der ersten Zeile einer Stapeldatei – am Ende wird das Echo automatisch wieder eingeschaltet. Noch schöner wäre dieser Batch, wenn er nicht nur auf eine Datei anwendbar wäre, sondern auf beliebige. Dazu geben wir den Dateinamen als Parameter beim Aufruf an. In der Batchdatei steht dafür ein Platzhalter. Die Parameter (bis zu neun Stück) werden mit %1 bis %9 bezeichnet. Der gleiche Batch wie oben für beliebige Dateien sieht so aus:

```
if exist %1 type %1
if not exist %1 echo %1 ist nicht vorhanden
```

Der ECHO-Befehl wird – wie bereits gezeigt – entweder dazu genutzt, die Ausgaben der Befehle zu unterdrücken (ON/OFF) oder aber um Meldungen auf den Bildschirm zu bringen. Der Aufruf der angenommenen Datei TEST.BAT erfolgt dann mit

```
test dateiname
```

Sie listet als **test autoexec.bat** entweder die Datei oder schreibt, daß sie nicht vorhanden ist. Als nächstes soll verhindert werden, daß man auch die CONFIG.SYS damit ansehen kann. Alle anderen Dateien ja, aber die Konfigurationsdatei soll nicht zugänglich sein. Dazu sollte es genügen, das Kommando

```
if %1 == config.sys echo Diese Datei ist tabu
```

voranzustellen (beachten Sie bitte, daß es sich um zwei Gleichheitszeichen handelt). Beim Aufruf von **test config.sys** werden Sie feststellen, daß die Ausschrift wirklich kommt, die Datei aber anschließend trotzdem gezeigt wird (sofern sie vorhanden ist). Ursache dafür ist, daß alle Befehle nacheinander durchlaufen werden. Um dem zu begegnen, kann mit **GOTO** verzweigt werden, indem eine Sprungmarke angegeben wird. Sie kann beliebig lang sein, aber nur 8 Zeichen werden zur Auswertung herangezogen. Die Marke selbst wird von einem Doppelpunkt eingeleitet. Wird sie nicht gefunden, so bricht DOS mit einer Fehlermeldung ab. Das Beispiel von oben sieht dann richtig so aus:

```
echo off
if %1 == config.sys goto tabu
if exist %1 type %1
if not exist %1 echo %1 ist nicht vorhanden
exit
:tabu
echo Diese Datei ist tabu.
```

Beim Dateinamen (wie bei allen Parametern) muß Groß- und Kleinschreibung unterschieden werden.
Als Beispiel für die Übergabe mehrerer Parameter schreiben wir uns jetzt eine Befehlsdatei, die mehrere Dateien listet:

```
if exist %1 type %1
if exist %2 type %2
if exist %3 type %3
```

Beim Aufruf mit

```
test config.sys autoexec.bat brief.txt
```

werden alle drei Dateien nacheinander angezeigt. Damit können sie umgehen, daß beim TYPE-Kommando keine Joker erlaubt sind. Trick am Rande: Zum Listen mehrerer Dateien kann auch

```
copy dateien con
```

verwendet werden, beispielsweise

```
copy *.bat con
```

zum Listen aller BAT-Dateien. Wenn Ihnen das nicht einleuchtet: Einmal aussetzen und zwei Folgen zurück zum COPY-Befehl.
Wie bereits erwähnt, können die Parameter %1 bis %9 verwendet werden (%0 enthält den Namen der BAT-Datei selbst). Sollten Sie wirklich einmal nicht damit auskommen, so kann der **SHIFT**-Befehl zur Anwendung kommen. Er verschiebt die Parameterkennungen nach links. Sie erreichen unter %1 dann den zweiten Parameter und unter %9 den zehnten. Durch mehrmaliges Anwenden können so alle Parameter zugänglich gemacht werden. Für die meisten Anwendungen wird das aber kaum nötig sein.
Unbefriedigend beim Anzeigen der drei Dateien ist, daß stets genau drei Parameter angegeben werden müssen. Besser wäre es, wenn bei einem weggelassenen Parameter die Aktion nicht ausgeführt wird. Dazu wird die Parameterzeichenkette daraufhin geprüft, ob sie leer ist, und zwar mit **if %1* == * ...** Eine leere Zeichenkette im eigentlichen Sinne ist nicht vorhanden. Deshalb wird ein beliebiges Zeichen angehängt (hier ein Stern) und verglichen, ob es sich nur um einen Stern handelt. Zur besseren Lesbarkeit ist aber auch **if "%1" == "" ...** möglich. Der Batch zum Anzeigen von 1 bis 3 Dateien hat dann folgendes Aussehen:

```
if exist %1 type %1
if "%2" == "" goto ende
if exist %2 type %2
if "%3" == "" goto ende
if exist %3 type %3
:ende
```

Wenn Sie genau aufgepaßt haben, werden Sie feststellen, daß jetzt doppelt geprüft wird. Denn ein nicht übergebener Parameter wird als leere Zeichenkette interpretiert und deshalb als Datei nicht gefunden, was auch keine Fehlermeldung auslöst. In vielen Fällen ist es aber wichtig, auf die Existenz eines Parameters zu prüfen. So würde die Anweisung **format %1:** beim Fehlen des Laufwerksparameters **format :**ergeben. Wird die Existenz des Para-

CALL datei [parameter]	Aufruf einer weiteren Batch-Datei (ab DOS 3.30)
GOTO marke :marke	Verzweigen zu einer Sprungmarke
ECHO [ON | OFF | meldung]	Ausgabe von Meldungen auf den Bildschirm
EXIT	Vorzeitiges Beenden einer Batch-Datei
FOR %%variable IN (parameter) DO kommando	Wiederholte Abarbeitung von Kommandos, solange die Bedingungen erfüllt sind (Bei direkter Abarbeitung nur ein Prozentzeichen)
IF [NOT] bedingung kommando	Bedingte Ausführung von Kommandos
bedingung: ERRORLEVEL n	Rückgabewert größer oder gleich n
EXIST dateiname	Vorhandensein einer Datei
string1 == string2	Gleichheit von Zeichenketten
PAUSE [meldung]	Warten auf Tastendruck
REM kommentar	Kommentare in Batch-Dateien (keine Wirkung)
SHIFT	Verschieben der Parameter nach links

meters getestet, ergibt sich die Möglichkeit, einen Standard (z. B. a:) anzunehmen.
Nicht unterstützt von der DOS-Kommandosprache werden Tastatureingaben, die dem Nutzer die Auswahl aus einem Menü oder die Beantwortung einer Abfrage (etwa: „Wirklich Formatieren (J/N)") gestatten. Deshalb wird zu einem Trick gegriffen, den wir – obwohl häufig in Publikationen gezeigt – wegen seiner Nützlichkeit kurz vorstellen möchten. Diesem liegt die Tatsache zugrunde, daß mit **ERRORLEVEL** der Programmrückkehrcode im Prozessorregister AL ausgewertet wird. Beispielsweise gibt Format (wie den DOS-Unterlagen zu entnehmen ist) eine 0 für fehlerfreie Beendigung der Arbeit, eine 3 für einen Abbruch mit <CTRL> <C> bzw. <Strg> <C> und eine 4 für einen Diskettenfehler zurück. Das läßt sich mit folgendem Batch-Programm ausprobieren:

```
format a:
if errorlevel 4 goto fehler
if errorlevel 3 goto abbruch
if errorlevel 0 goto okay
```

usw.
Um Tastatureingaben auszuwerten, wird nur ein kleines Programm benötigt, daß den Tastendruck abfragt und ASCII-Dode übergibt. Am einfachsten wird folgendes Assemblerprogramm mit einem Editor erfaßt und durch den Debugger Debug (befindet sich unter den externen DOS-Kommandos) geschickt. Das ist auch für Nutzer mit wenig Erfahrung zu schaffen. Hier das Programm TASTE.ASM:

```
a
mov ah,0
int 16h
mov ah,4ch
int 21h
        (Leerzeile)
r cx 8
n taste.com
w
q
```

Übersetzt wird es durch
debug < test.asm
und das fertige Programm TASTE.COM kann dazu verwendet werden, den Tastencode als ERRORLEVEL zurückzugeben. Es gibt sicher elegantere Lösungen, aber keine kürzere. Sie brauchen nur noch in der ASCII-Tabelle den Zeichencode nachzusehen. Beachten Sie dabei, daß Tasten, die einen erweiterten Code abgeben (Funktionstasten, Kombinationen mit <ALT> oder <CTRL>) keine Verwendung finden können. Das eigentliche Ziel bei der Einführung des ERRORLEVEL war, auf eine fehlerhafte Beendigung eines Programms reagieren zu können. Als Beispiel wird häufig das Assemblieren von Programmen gezeigt und je nachdem, ob das fehlerfrei geschieht oder nicht, wird zum Linken oder zum erneuten Editieren verzweigt. Wir benutzen den ERRORLEVEL-Ausdruck also wieder einmal zweckentfremdet (wie schon den Prompt-Befehl). Dazu wird ein kleines Menü erzeugt, aus dem der Anwender ein Programm aussuchen kann, das gestartet werden soll (in diesem Falle Wordstar, dBase und Multiplan). Die Auswahl erfolgt durch Eingabe des Anfangsbuchstabens (Programm Taste). Und so könnte eine kleine Batch-Datei aussehen:

```
echo off
echo Aufruf von Programmen
echo.
echo w – Wordstar
echo m – Multiplan
echo d – dBase
echo.
echo Bitte Auswahl:
taste
if errorlevel 119 goto ws
if errorlevel 109 goto mp
if errorlevel 100 goto db
exit
:ws
c:\text\ws
exit
:mp
c:\calc\mp
exit
:db
c:\data\dbase
```

Aus Platzgründen wurde auf jegliche Verschönerungsmaßnahmen verzichtet. Statt des einfachen Programmstarts sollten noch die Standarddirectories eingestellt, Suchpfade festgelegt und eine nette Ausschrift gegeben werden.
Mit **echo.** wird einfach eine Leerzeile erzeugt. Und nun die Auswertung: **if errorlevel 119** testet, ob ein Code größer (!) oder gleich 119 übergeben wurde. Die 119 ist gerade ein kleines w in der ASCII-Tabelle. Dieses w bewirkt dann den Start von Wordstar. Anschließend wird auf m (109) und d (100) geprüft. Wichtig ist die Aussage, daß stets auf größer oder gleich untersucht wird. Das hat den Hintergrund, daß beim Test auf Fehler nicht alle möglichen Codes abgefragt werden können. Ein fehlerfreies Programm gibt 0 zurück, und mit aufsteigenden Nummern werden die Fehler schwerwiegender. Auf diese Weise müssen zuerst die schweren Fehler bearbeitet werden, z. B.

```
if errorlevel 8 goto abbruch
if errorlevel 4 goto warnung
if errorlevel 0 goto okay
```

So wird bei allen schlimmeren Fehlern als 8 abgebrochen, bei Fehler zwischen 4 und 8 gewarnt usw.
Für unser obiges Beispiel heißt das aber, daß bei allen Tasten mit einem Code größer als 119 zu Wordstar verzweigt wird. Also zum Beispiel, wenn Sie ein z eingeben. Und bei einem p gehen Sie zu Multiplan – ein kleiner Nachteil dieses kurzen Programms. Ein oft begangener Fehler bei dieser Art der Batch-Programmierung ist, daß die Prüfung mit ERRORLEVEL nicht in absteigender Reihenfolge geschieht. Dann landen Sie stets im ersten Menüpunkt.
Wenn Ihnen der Umgang mit dem Programm TASTE zu schwierig erscheint, so können sie auch ASK aus den Norton-Utilities benutzen. Das oben gezeigte Programm sieht dann (leicht gekürzt) so aus:

```
echo w – Wordstar
echo m – Multiplan
echo d – dBase
ask "Bitte Auswahl", dmw
if errorlevel 3 goto ws
if errorlevel 2 goto mp
if errorlevel 1 goto db
```

usw.
Dem Programm ASK werden neben einer Ausschrift die Anfangsbuchstaben der möglichen Antworten übergeben, und man erhält die Nummer des gedrückten Buchstabens zurück (hier nicht der ASCII-Code). Weiterhin muß aber in absteigender Reihenfolge getestet werden.

Literatur

/1/ MS-DOS-Referenzkarte. Mikroprozessortechnik, Berlin 2 (1988) 12, 3. US

/2/ Smode, D.: Das große MS-DOS-Profi-Arbeitsbuch. München: Franzis-Verlag GmbH 1987

In der nächsten Folge bleiben wir noch ein wenig bei der Stapelverarbeitung, und Sie erfahren dann, was es mit Filtern und Pipes in DOS auf sich hat und daß Umleitungen nützlich sein können.

Im Bild 10 ist ein Demonstrationsprogramm für den Algorithmus 4 zu sehen. Die Algorithmen 1 bis 3 sind daraus leicht ableitbar, indem die Prozedur *zeige* entsprechend verändert wird. Es wurde bewußt auf eine Laufzeitoptimierung verzichtet, um das Programm übersichtlich zu gestalten.
Anmerkung: Alle Abbildungen sind Kopien des Bildschirms, die dadurch im Maßstab leicht deformiert werden. Insbesondere ist bei der Bildschirmdarstellung die gefüllte Fläche stets ein Quadrat, und in Bild 9 liegt ein Rechteck und kein Parallelogramm vor.

Ausblick

Die hier vorgestellten Algorithmen sind in verschiedener Weise ausbaufähig. Es wäre beispielsweise denkbar, daß das Originalbild nicht durch Geraden analysiert wird, sondern durch andere geometrische Gebilde (z. B. Pixelquadrate). Dabei ist stets mit steigenden Zeiten für die Bildanalyse zu rechnen. Außerdem ist auch zu klären, wie diese geometrischen Grundformen transformiert werden können. In diesem Sinne sind die vorgestellten Algorithmen ein Kompromiß zwischen Geschwindigkeit und Qualität der Bildbearbeitung. Eine Beschleunigung aller Algorithmen ist weiterhin denkbar durch die Ausnutzung der speziellen byteweisen Speicherung des Bildes. Dementsprechend ist nicht eine pixelorientierte Bildanalyse und -bearbeitung, sondern eine byteorientierte genauso denkbar. Allerdings wäre hier der programmtechnische Aufwand beachtlich hoch.

Literatur

/1/ Pavlidis, T.: Algorithmus for Graphics and Image Processing. Computer Science Press, Inc., Rockville 1982
/2/ Angell, I. O.: A practical introduction to computer graphics. The Macmillan Press 1981
/3/ TURBO-PASCAL, Referenzhandbuch der Version 4.0. Heimsoeth & Borland 1987

KONTAKT

Akademie der Landwirtschaftswissenschaften, Institut für Getreideforschung Bernburg-Hadmersleben, AG Modellierung, Krausenstraße 38/39, Berlin, 1086; Tel. 2 00 09 51

TERMINE

4. Tagung Elektronik-Technologie

WER? Humboldt-Universität, Sektion Elektronik
WANN? 5. bis 8. November 1990
WO? Berlin, Kongreßhalle am Alexanderplat
WAS?
- Technologie mikro- und optoelektronischer Systemelemente
- CAD/CAM
- Prozeßautomatisierung
- Bildverarbeitung
- Meß- und Sensortechnik
- optische Nachrichtentechnik
- Rechnerkommunikation
- Informatik in der Leheraubildung

WIE? Anfragen richten Sie bitte an: Humboldt-Universität Berlin, Sektion Elektronik, Dr. Hochmuth, Invalidenstraße 110, 1040 Berlin *Dr. Härtig*

kennengelernt

Quick-Basic 4/4.5

Dr. Knut Löschke

Neben Cobol und Fortran ist Basic die wohl am meisten benutzte Programmiersprache. Durch Homecomputer und PCs verbreitet, erfreut sie sich weiter großer Beliebtheit, obwohl die Nachteile der alten Basic-Interpreter-Sprache unübersehbar sind. Als Gründe für die prinzipielle Ablehnung werden insbesondere die mangelnde Strukturierbarkeit und – in deren Gefolge – die Probleme der Projektentwicklung und Softwarewartung ins Feld geführt.

Solche Fragen scheinen durch die Sprache Pascal beantwortet und deren „Siegeszug" durch Borlands Turbo-Pascal unangefochten zu sein.
In den letzten Jahren wurden jedoch von verschiedenen großen Softwarehäusern neue Basic-Varianten angeboten. Sie treten mit dem Anspruch an, die Nachteile des alten Basic beseitigt, alle Vorteile bewahrt und weiter ausgebaut zu haben und eine Alternative auch für professionelle Softwareentwicklung zu bieten.
Ein spannendes Kopf-an-Kopf-Rennen lieferten sich dabei Borland mit Turbo-Basic und Microsoft mit Quick-Basic. Inzwischen wird anerkannt, daß Microsoft mit Quick-Basic 4.5 die Nase vorn hat, und Bergland entwickelt Turbo-Basic nicht weiter.

Das Grundkonzept von Quick-Basic

Quick-Basic ist weder ein Interpreter noch ein Compiler schlechthin. Mit Quick-Basic 4 und 4.5 legt Microsoft ein qualitativ neues Software-Entwicklungskonzept vor, das als Linie in allen anderen Microsoftsprachen (Quick-C, Quick-Pascal) jetzt und zukünftig verfolgt werden soll. Quick-Basic 4.5 bietet wohl als Flaggschiff all das, was darunter zu verstehen ist.
Sichtbare Äußerung des Konzeptes ist die ähnlich gestaltete Entwicklungsumgebung aller neuen Microsoft-Sprachen (Quick-...), das gleichartige Bibliothekskonzept (QuickLib), die damit verbundene Mischsprachenprogrammierung (Objekt-Module aller Sprachen können miteinander gelinkt werden), und die Verwendung des Debuggers Codeview für alle für ihn erzeugten Programme.
Nicht ohne tieferen Hintersinn sagt Microsofts Produktmanager Greg Lobdell: „Basic wird sich zu der Sprache entwickeln, in die die Microsoft-Anwendungen eingebettet sind". Mit Quick-Basic wird sowohl dem Hobby-Programmierer als auch dem professionellen Softwareentwickler ein Werkzeug zur Verfügung gestellt, daß kaum noch Wünsche offen läßt und hochmütige Ignoranz solchen Erneuerungen gegenüber in die Ecke stellt.

Die Entwicklungsumgebung Quick-Basic 4/4.5

Eine Sprache kann noch so stark und ausgefeilt sein – moderne Softwareentwicklung mißt sie nicht ohne Grund an den mit ihr zur Verfügung gestellten Entwicklungsumgebungen und Werkzeugen. In einer modernen, menügeführten Benutzeroberfläche integriert Quick-Basic (QB) einen leitungsfähigen, word-ähnlichen Texteditor, die Softwareentwicklungsunterstützung, den Compiler, ein Hilfe-System und Debugger-Werkzeuge.
Das klingt alles noch nicht sehr revolutionär, ist man von Turbo-Pascal her verwöhnt. Verblüffung erzeugt QB aber spätestens dann, wenn man beginnt ein Programm einzutippen. Jetzt wird deutlich, was Softwareentwicklungsunterstützung gegenwärtig heißen kann.
Weil QB jede eingegebene Zeile sofort compiliert, wird – soweit möglich – die Syntax geprüft, Leerzeichen werden korrigiert und die Schreibweise der Variablen übernommen. Das Ergebnis dieses blitzartigen Vorgangs (150 000 Zeilen sollen pro Minute compiliert werden können ...) sieht der Programmierer z. B. entweder in einer Fehlernachricht oder in der Großschreibung aller Schlüsselwörter und in der Korrektur der Schreibweise aller gleichbenannten Variablen im Geltungsbereich einschließlich der Leerzeichenanordnung.
Eine vorbildliche Unterstützung bietet bereits das Hilfe-System von QB 4. Die Version 4.5 geht noch darüber hinaus.
Neben der *manuellen Durchforstung* der klar strukturierten Hilfstexte bietet QB gewissermaßen online per Knopfdruck Hilfe zu dem Sprachkonstrukt an, auf dem der Cursor steht. Aber nicht nur Hilfe schlechthin, sondern ganze Beispielprogramme, Prozeduren, nützliche Querverweise auf Ähnliches usw. Das alles kann sogar bei Bedarf in den Programmtext übernommen oder probeweise in einem gesonderten Interpretermodus abgearbeitet werden. In diesem Direktmodus, der in einem eigenen Fenster aktiviert wird, kann der Entwickler ganze Programmteile oder auch einzelne Zeiten testen, bevor er sie in den eigentlichen Quelltext aufnimmt. Hilfreich ist das zum Beispiel auch beim separaten Rechnen (dezimal oder hexadezimal). Es kann aber auch benutzt werden, um *schnell einmal* eine Datei zu eröffnen, einen Satz auszulsesen und dessen Inhalt auf dem Bildschirm anzusehen.
Die Projektentwicklung mit QB wird durch ein Modulkonzept unterstützt, das in verschiedenen Ebenen wirksam wird, wobei einzelne Niveaus weggelassen werden können.
Ein Projekt besteht für QB aus einer Hierarchie von Quelltext-Strukturen. Die Wurzel ist das Hauptprogramm. Es besteht aus Modulen, die für sich wiederum lauffähige Programme sein können. Mit QB können diese einzeln editiert, gespeichert und getestet werden.
Als Unterprogramme fungieren SUB-Routinen (Prozeduren) und Funktionen.
Programme der Wurzel-Ebene können natürlich andere – sogar mit Übergabe von Variablen – starten (CHAIN) oder als Kindprozeß (SHELL) aufrufen. Module können, weil sie im RAM verwaltet werden, gegenseitig Unterprogramme aufrufen.
Eine weitere Möglichkeit bildet die Bibliotheksebene. QB verwaltet einen Bibliothekar, der es gestattet, eigene Objektmodule oder Module anderer Microsoft-Programmiersprachen (Assembler, C, Pascal, Fortran) einzubinden. Letztlich wird damit die Sprache beliebig erweiterbar (der Aufruf der Module erfolgt praktisch genauso, als ob Basic-Befehle benutzt werden). Erst beim letztlichen Ablegen eines EXE-Files werden die Module und die Bibliotheksroutinen miteinander gelinkt. Ein Systemfile sorgt dafür, daß zusammengehörige Module eventuell gleich gemeinsam unter dem Namen des Projektes geladen werden. Die Verwendung von Unterprogrammen und Funktionen, die auch rekursiv benutzt werden können, unterstützt QB auf besondere Weise. Zunächst kann an beliebiger Stelle im Quelltext SUB oder FUNCTION geschrieben werden, und man befindet sich sofort nur noch innerhalb dieses Quelltext-Abschnittes. Wird in den Text des Moduls zurückgegangen, ist das Unterprogramm dort als Quelle überhaupt nicht zu finden. QB hat sie nämlich inzwischen säuberlich in eine gesondert verwaltete, alphabetische Liste aller Module und ihrer Unterprogramme aufgenommen und am Modulbeginn deklariert (DECLARE). Auf diese Weise können mit einem Knopfdruck ganze Unterprogramme gelöscht oder von Modul zu Modul transportiert werden.
Aber nicht nur das. Beim jetzt möglichen Aufruf werden noch beim Schreiben die Typen, die Reihenfolge und die Anzahl der zu übergebenden Parameter geprüft und die Nichtübereinstimmung mit der Deklaration gemeldet.
Module selbst müssen explizit verlassen oder angewählt werden. Es ist so, daß QB diese Strukturen mit globalen und lokalen Variablen bedient und nutzerdefinierte Variablentypen (TYPE) zuläßt, die Überschaubarkeit und Handhabbarkeit gewährleisten.
Alle genannten Features zwingen noch nicht zum

strukturierten Programmieren – machen es aber so leicht und bequem, daß man es praktisch von ganz alleine tut. Und top-down dazu.
Ein weiterer Pluspunkt sind die ausgebauten Debugging-Fähigkeiten. Abgesehen von der Möglichkeit, am fertigen EXE-Programm mit Codeview ansetzen zu können, erlaubt QB praktisch alles, was an Break- und Watch-Punkten, an Variablenverfolgung usw. denkbar ist. Weil QB 4/4.5 eigentlich ein Interpreter ist, kann er aber auch alles, was einen solchen auszeichnet: Verlangsamung des Ablaufs, schrittweises Abarbeiten, Sprünge zu beliebigen Programmstellen, sogar eine Rückwärts-Abarbeitung ist möglich...

Die Sprache Quick-Basic

Microsoft behauptet kühn, daß QB zu GW-Basic (BASICA) kompatibel sei. Das stimmt auch in fast allen Fällen. Dennoch kann es zu Problemen kommen, wenn Interpreter-Basic-Programme unter QB bearbeitet werden sollen. Sie sind jedoch dank des Hilfesystems meist schnell lösbar (d. h. an die etwas andere Sprachkonstruktion anpaßbar). Wer Zeilennummern nun nicht mehr mag, kann sie sich mit Hilfe eines Dienstprogramms entfernen lassen. QB braucht sie nicht; sie stören aber auch nicht.
Gegenüber GW-Basic wurde QB – beginnend mit der Version 2 bis zur 4.5 – um moderne Strukturelemente kontinuierlich erweitert (CASE-Konstrukt, IF-ELSEIF-ELSE-ENDIF, DO WHILE – LOOP UNTIL sowie FUNCTION und SUB in Verbindung mit globalen, lokalen und typisierten Variablen (TYPE), rekursiven Prozeduraufrufen usw.). Hinzu kamen auch viele nützliche Befehle oder Erweiterungen derselben. Weiterhin stehen die leistungsfähigen Grafik-Befehle nun für CGA-, EGA-, VGA- und Herkuleskarten zur Verfügung. Neben den 2-Byte-INTEGER-, SINGLE-(7 Stellen) und DOUBLE-(15 Stellen)Variablen gibt es nun auch eine 4-Byte LONG INTEGER und numerische Konstanten. Dem dynamischen String-Typ wurde eine Stringvariable konstanter Länge hinzugesellt. Ein kaum zu überschätzender Vorteil sind dynamische Felder (REDIM). Ab QB 3 wird der mathematische Koprozessor unterstützt und intern ab der Version 4 im IEEE-Format bedient.
Standardmäßig stehen sowohl für Strings als auch für numerische Variablen je rund 50 KByte Speicher zur Verfügung. Mit einem kleinen Trick ist es aber möglich (Strings als eindimensionale Felder; jedes Feld mit konstanter Länge), den Stringspeicher bis zur maximal nutzbaren RAM-Größe auszudehnen, weil für jedes Stringfeld konstanter Länge 64 KByte reserviert werden.
Der Extended-Memory-Bereich eines ATs ist leider nicht direkt nutzbar.

EXE-Files

Beim Laden eines Quelltextes bearbeitet QB ab Version 4 ihn so (loading and parsing), daß eine vorkompilierte Quelle im Speicher steht (Threated-P-Code). Jede weitere oder eingefügte Programmzeile wird sofort in dieses Format überführt. Nach dem Programmstart entfällt damit ein zusätzlicher Compile-Vorgang und das Programm ist *sofort da*. Bei sehr langen Quellen und vor allem bei sehr vielen Prozeduren kann ein merklicher (im 1 bis 4 Sekundenbereich liegender) Binding-Prozeß auftreten. Im Gegensatz zu allen anderen vergleichbaren Entwicklungssystemen, befindet sich im Speicher von QB nur eine Form des Programms, der P-Code, aus dem sowohl der explizite Quelltext als auch das ausführbare Programm im Bedarfsfall zeilenweise erzeugt wird.
Im Laufzeitverhalten und in sehr seltenen Fehlersituationen wird deutlich, daß QB damit ein kompilierender Interpreter oder interpretierender Compiler ist. Letztlich erzeugte EXE-Files sind dann auch noch schneller als ihre Vorstadien in der Entwicklungsumgebung.
Soll nach ausführlicher Testung das Programm zum selbständigen EXE-File werden, ruft QB einen separaten Basic-Compiler und den Microsoft-Linker (LINK) auf. Das erscheint zunächst umständlich, liegt doch eine vorkompilierte Quelle im Speicher vor. Sinnvoll wird die ganze Sache aber dadurch, daß ja jeder Compiler vom Nutzer an QB angebunden werden kann. Also auch der Basic-Compiler 6.0 von Microsoft, mit dem OS/2-Programme erzeugt werden können.

```
Programm 1
t = TIMER
X% = 0
Y% = 9
FOR i% = 1 TO 30000
     X% = X% + (Y% * Y% - Y%) \ Y%
NEXT
PRINT TIMER - t

Programm 2
t = TIMER
X! = 0!
Y! = 9.9
FOR i% = 1 TO 20000
     X! = X! + (Y! * Y! - Y!) / Y!
NEXT
PRINT TIMER - t

Programm 3
t = TIMER
X! = 0.0
Y! = 9.9
FOR i% = 1 to 5000
     X! = X! + SIN(ATN(COS(LOG(Y!))))
NEXT
PRINT TIMER - t

Programm 4
t = TIMER
FOR i% = 1 to 1000
     PRINT "1234567890qwertzuiop", i%
NEXT
PRINT TIMER - t

Programm 5
SCREEN 2
t = TIMER
FOR i% = 1 to 100
     FOR j% = 1 to 100
          PSET (I%, j%)
     NEXT j%
NEXT i%
PRINT TIMER - t

Programm 6
t = TIMER
OPEN "test.dat" FOR RANDOM AS #1 LEN = 20
FIELD #1, 20 AS A$
FOR I% = 1 to 1000
     A$ = "1234567890qwertzuiop"
     PUT #1, I%
NEXT
CLOSE #1
KILL "test.dat"
PRINT TIMER - t
```

Bild 1

Auffällig ist die beachtliche Größe der EXE-Files. Wird das Laufzeitmodul mit eingebunden (Standalone-Programme), so kommt man kaum unter 20 KByte. Allerdings kennt QB keine Größenbegrenzung nach oben.
Die Länge der Programme kann eventuell verschmerzt werden, zeigen sie dann, welche Geschwindigkeiten sie erreichen. Sicher sind Benchmark-Tests so eine Sache. QB schneidet aber gegenüber allen Compiler-Basic-Versionen im Durchschnitt als Sieger ab. Selbst Turbo-Pascal 5.0 hinkt in meinen Tests (Bild 1) vor allem in der Arithmetik hinterher... (Tafel 1). Ihre Umsetzung in Turbo-Pascal 5.0 erfolgte mit höchstmöglicher Ähnlichkeit.

Toolboxen und Umfeld

Wie eingangs erwähnt, entscheidet sich ein professioneller Programmierer für eine Programmiersprache aus viel mehr Gründen denn aus Gründen des Sieges bei Geschwindigkeitstest. QB muß sich sowohl gegen die (jetzt unbegründeten) Vorurteile gegen Basic als auch gegen ein weit ausgebautes Netz von Toolboxen, Modulen und Units anderer Entwicklungsumgebungen durchsetzen. Unter diesem Gesichtspunkt sind Turbo-Pascal und Microsoft-C sicher unangefochtene Favoriten.
QB ist derzeit leider nur unter MS-DOS und auf Apple MacIntosh lauffähig.
Allerdings kommt QB Basic-Programmierern entgegen, die zu MS-DOS umsteigen und bietet für etwa 340 DM zahlreiche Vorteile, die man in anderen Programmiersprachen unter MS-DOS derzeit vergeblich sucht.

Tafel 1

Programm	Quick-Basic 2	Quick-Basic 4	Turbo-Basic 2.0	Turbo-Pascal 5.0
1	0,441	0,378	0,439	0,445
2	6,641	1,980	2,363	5,998
3	16,148	4,558	4,228	6,598
4	34,328	12,027	17,848	10,392
5	1,648	1,308	1,593	1,480
6	1,867	1,980	1,812	1,980

Anmerkung: Benchmark-Programme nach Personalcomputer-World, getestet auf einem PC AT, 80286 mit Koprozessor 80287, EGA-Grafik und Festplatte; Zeitangaben in Sekunden.

TERMINE

1. Werkstattag des Computerclubs

WER? Kammer der Technik, Bezirksvorstand Frankfurt (O).
WANN? 24. Oktober 1990
WO? Franfurt (O).
WAS?
- Systemprogrammierung (DOS-Rufe, MASM, TASM, Turbo-Pascal 5.0)
- Datensicherheit
- Einsatz von Druckern

WIE? Nähere Informationen sind erhältlich beim Bezirksvorstand der KDT Frankfurt (O)., Ebertusstraße 2, Postschließfach 110, Frankfurt (O)., 1200; Tel. 36 93 30 *Hamhalter*

Forth-Workshop

WER? Kammer der Technik Suhl
WANN? 7. bis 9. November 1990
WO? Suhl
WAS?
- Steuerungsaufgaben
- Aufgaben der KI
- Forth-Einsatz in der Informatikausbildung

WIE? Anmeldungen richten Sie bitte an: Kammer der Technik Suhl, Interessengemeinschaft Forth, „Workshop", Postfach 190, Ilmenau, 6300 *Dr. Finsterbusch/Dr. Horn*

Kopierbahnhof VGA-RAM

Verwendung des VGA-RAMs jenseits A000:9600H.

Jürgen Petsch

Im PC herrscht in aller Regel chronischer Speicherplatzmangel. Um so verwunderlicher ist es, daß viele Besitzer von VGA-Karten ungewollt über 100 KByte ungenutzt lassen müssen.
Eine kleine Turbo-Pascal-Unit zeigt Ihnen, wie Sie dieser Verschwendung für einige Anwendungen Einhalt gebieten.

Eine IBM-kompatible VGA-Karte ist standardmäßig mit 256 KByte bestückt, die im häufig verwendeten Modus 12H (640 × 480 Pixel bei 16 Farben) in 4 Ebenen à 64 KByte aufgeteilt sind. Für den Bildspeicher werden aber nur 38 400 Byte in vier Ebenen, also 153 600 Byte benötigt. Die ungenutzten 108 544 Byte (27 136 je Ebene) warten nur auf sinnvollen Einsatz.
Der angezeigte Bildspeicher beginnt nach der Initialisierung bei der Adresse A000:0000H. Solange man diese Startadresse nicht absichtlich verändert, kann man den Bereich ab A000:9600H prima als Spielwiese für eigene Ideen verwenden. Natürlich könnte man den Platz zur zeitweiligen Ablage von Datenfeldern nutzen, wenn man sich dazu vom VGA-Chip die entsprechende Ebene zuteilen läßt (Map Mask Register für das Schreiben, Read Map Select Register für das Lesen).

Musketiere, 32 Bit breit

Wie einfallslos! Es ist doch viel naheliegender, den ungenutzten Speicher zur Ablage von Bildausschnitten zu verwenden. Dabei unterstützt der *Write Mode 1* des VGA-Chips tatkräftig das Kopieren von Speicherbereichen im Segment A000H. Dies läuft dann gänzlich innerhalb des VGA-RAMs ab.
Mit einer Leseoperation wird das 32-Bit-Latchregister mit dem Inhalt einer vorgegebenen Speicherzelle aus den vier Ebenen geladen und mit einem nachfolgenden Schreibbefehl in die vier Ebenen an eine andere Stelle zurückgeschrieben. So transportiert ein einzelner Assemblerbefehl (MOVSB) zum Laden eines Bytes 32 Bit mit einem Schwung. Auf ähnliche Weise ist es im *Write Mode 2* möglich, mit einem STOSB acht nebeneinanderliegenden Pixeln die gleiche Farbe zu verleihen.
Als praktische Anwendung dienen die beiden von Turbo-Pascal 4.0/5.x aufrufbaren Prozeduren SaveSprite und RestSprite. Sie nutzen den *Write Mode 1* und erlauben das Retten und Restaurieren von rechteckigen Bildausschnitten. Um solche Bildausschnitte mit einer vorgegebenen Farbe zu füllen, bedient sich FillSprite der zahlreichen Möglichkeiten des *Write Mode 2*.

Drei Weise...

Alle drei Prozeduren sind im Quelltext SPRIASM.ASM enthalten. Der Deklarationsteil in SPRI.PAS stellt das Interface zu Turbo-Pascal her, der die Unit SPRI vervollständigt. Sie ist genau wie die bereits in einer früheren Ausgabe der c't [1] vorgestellte Unit VGA als Stand-alone-Unit nutzbar. Die beiden Units lassen sich allerdings auch vereinen.

Die drei Musketiere, Save-, Fill- und Rest-Sprite, verleihen dem Antlitz selbstgeschriebener Programme Professionalität. So sind sie zum Beispiel in der Lage, ein Fenster mit Hilfstexten zur Bedienung eines Programms in eine grafische Darstellung einzublenden. Beim Ausblenden geht der ursprüngliche Inhalt nicht etwa verloren, sondern RestSprite restauriert die vorher geretteten Pixel (Beispielprogramm: WINDEMO.PAS).
Die Anzahl der Fenster hängt natürlich von deren Größe ab. Schließlich benötigt auch der wiederherzustellende Bildbereich noch mal den Speicherplatz des Fensters, und gut 27 KByte sind schnell verbraucht.
Die beiden in Turbo-Pascal enthaltenen Prozeduren GetImage und PutImage können Sie natürlich auch dazu verwenden. Allerdings haben die hier vorgestellten Routinen eben den Vorteil, zur Ablage der Bildausschnitte keinen Platz im Hauptspeicher zu binden. Gerade dem ungenutzten Bereich des VGA-RAMs verhelfen unsere drei Routinen zu neuen Ehren. Außerdem transportieren SaveSprite und RestSprite die Daten auf einem 8-MHz-AT etwa 15- beziehungsweise 30mal schneller als Put- und GetImage. Kleines Opfer: Die vorgestellten Routinen gestatten keine Wahl der Übertragungsart (XOR, OR, AND usw.).

... aus dem VGA-Land

Die größere Geschwindigkeit erlaubt noch eine andere Anwendung. Das Demopro-

```
10 UNIT SPRI;  (* Turbo-Pascal 4.0/5.x
11                 (w) 1990 Jürgen Petsch
12                 (c) 1990 Heinz Heise Verlag GmbH & Co KG *)
13 INTERFACE
14   Procedure SaveSprite
15    (X,Y,Breite,Hoehe:Integer; StartAdr:Word);
16   Procedure RestSprite
17    (X,Y,Breite,Hoehe:Integer; Startadr:Word);
18   Procedure FillSprite
19    (X,Y,Breite,Hoehe,Color:Integer);
20
21 IMPLEMENTATION
22   {$L SPRIASM.OBJ }
23   Procedure SaveSprite; External;
24   Procedure RestSprite; External;
25   Procedure FillSprite; External;
26 END.
```

Bildauschnitte schnell kopieren – ohne DOS-RAM-Verbrauch: Diese UNIT ist der „heiße Draht“ zur VGA-Karte.

```
10 program SUNSHADE; (* Turbo-Pascal 4.0/5.x + UNIT SPRI
11                      (w) 1990 Jürgen Petsch
12                      (c) 1990 Heinz Heise Verlag GmbH & Co KG *)
13 (*$N+,E+,R+ *)
14
15 (* Zeigt die Verwendung des ungenutzten Bildschirmspeichers
16    ab der Adresse A000:9600H zur Darstellung von bewegten
17    Sprites mit Hilfe von SaveSprite, RestSprite und
18    FillSprite. Zeichnet ein Sonnenschirmchen, dreht und bewegt es.
19 *)
20
21 Uses CRT, GRAPH, SPRI;
22
23 Var
24   Treiber, Modus, I, J, K, X, Y,
25   Farbe, AdrOffs, Del, Hoehe, xx : Integer;
26   Ch                             : Char;
27   Alfa, Beta                     : Double;
28
29 Const
30   A=64;  B=32;          (* Grosse und kleine Halbachse *)
31
32 Begin
33   Treiber:=9; Modus:=2;
34   InitGraph (Treiber, Modus,'');
35   OutTextXY
36   (190,20,'c''t Demo, Mickeys Sonnenschirmchen fliegen lassen!');
37   OutTextXY
38   (190,30,'Erstmal 26 Bildchen zeichnen und verstecken.');
39   FOR I:= 0 TO 25 DO          (* 26 Bilder zeichnen *)
40   Begin
41     SetColor(1);
42     Line (A,B,A,B+30);        (* Senkrechte weisse Linie *)
43     Farbe:=10;
44     Alfa:= I*Pi/13;
45     FOR J:= 0 to 2 DO         (* 3 Fluegel zeichnen *)
46     BEGIN
47       SetColor (Farbe);
48       Beta:=0:
49       FOR K:= 0 TO 100 DO  (* Einen Fluegel zeichnen *)
50       BEGIN
51         BETA:= K*Pi/1000;
52         X:= Trunc ((A-2)*Cos(Alfa+Beta+2*J*Pi/3));
53         Y:= Trunc ((B-2)*Sin(Alfa+Beta+2*J*Pi/3));
54         Line (A+X,B+Y,A,B);
55       END;
56       Farbe:=Farbe+5;
57     END:
58     AdrOffs:=I*A Div 2*B;  (* Ein Bild ablegen *)
59     SaveSprite (0,0,A DIV 4,2*B,$9600+AdrOffs);
60     FillSprite (0,0,A DIV 4,2*B,0);   (* Bereich loeschen *)
61   END;
62   SetColor (15);
63   OutTextXY
64   (190,49,'Jetzt holen wir sie der Reihe nach wieder her.');
65   OutTextXY (212,80,'Beenden der Demo mit beliebiger Taste.');
66   Del:=500;                   (* Startverzögerung einstellen *)
67   Hoehe:=0;
68   Repeat
69     FOR I:= 0 TO 25 DO      (* 26 Bilder wieder holen *)
70     BEGIN
71       AdrOffs:=I*A Div 2*B;
72       RestSprite (200,400-Hoehe,A DIV 4,2*B,$9600+AdrOffs);
73       Delay (Del);
74
75       If (Del > 200) Then  (* Rotationsverzögerung *)
76         Del:= Del-20;      (* verringern          *)
77       If (Del > 30) then
78         Del:= Del-2;
79
80       If (Del < 40) and (Hoehe < 300) then
81         Hoehe:= Hoehe+1;
82     End;
83   Until Keypressed;
84   Ch:= ReadKey;
85   Ch:= ReadKey;
86 End.
```

Lassen Sie das Schirmchen nicht zu lange fliegen. Mickey und Minnie wären traurig, wenn es dann doch verloren ginge.

```
10  program WinDemo; (* Turbo-Pascal 4.0/5.x + UNIT SPRI
11                      (w) 1990 Jürgen Petsch
12                      (c) 1990 Heinz Heise Verlag GmbH & Co KG  *)
13  (*$R+ *)
14
15  (* Zeigt die Verwendung des ungenutzten Bildschirmspeichers
16     ab der Adresse A000:9600H zur Darstellung von Fenstern
17     mit Hilfe von SaveSprite, RestSprite und FillSprite *)
18
19  Uses CRT, GRAPH, SPRI;
20
21  Var
22    Treiber, Modus, I,
23    X,Y,Breite,Hoehe,Farbe              : Integer;
24    Startadr                            : Word;
25    Ch                                  : Char;
26    Text                                : String;
27
28 CONST      (* Lage und Groesse der Windows *)
29    X1= 80;  Y1=100;  B1= 40;  H1=200;   Text1 = 'WINDOW 1';
30    X2=200;  Y2=200;  B2= 50;  H2=160;   Text2 = 'WINDOW 2';
31    X3=120;  Y3=50;   B3= 30;  H3=300;   Text3 = 'WINDOW 3';
32
33 Procedure OpenWindow
34 (X,Y,Breite,Hoehe,Farbe:Integer; Startadr:Word; Text:String);
35   Begin
36     SaveSprite (X,Y,Breite,Hoehe,StartAdr);        (* Sichern *)
37     FillSprite (X,Y+16,Breite,Hoehe-16,Farbe);   (* Fuellen *)
38     FillSprite (X,Y,Breite,16,0);              (* Titelzeile *)
39     Rectangle (X,Y,X+8*Breite-1,Y+Hoehe-1);         (* Rahmen *)
40     Line (X,Y+16,X+8*Breite-1,y+16);
41     OutTextXY (X+20,Y+5,Text);
42   END;
43
44 Begin
45   Treiber:=9; Modus:=2;
46   InitGraph (Treiber, Modus,' ');
47   FOR i := 1 TO 15 DO                  (* Hintergrund zeichnen *)
48   BEGIN
49      SetColor (i);
50      Line (0,32*I,639,32*I);          (* quer *)
51      Line (41*I,0,41*I,479);          (* längs *)
52   END;
53   FillSprite (196,20,29,18,8);
54   OutTextXY (200,24,'Weiter mit beliebiger Taste');
55   Ch:= ReadKey;
56
57   OpenWindow (X1,Y1,B1,H1,9,$9600,Text1);
58   Ch:= ReadKey;
59   OpenWindow (X2,Y2,B2,H2,2,$9600+B1*H1,Text2);
60   Ch:= ReadKey:
61   OpenWindow (X3,Y3,B3,H3,6,$9600+B1*H1+B2*H2,Text3);
62   Ch:= ReadKey;
63
64   RestSprite (X3,Y3,B3,H3,$9600+B1*H1+B2*H2);
65   Ch:= ReadKey;
66   RestSprite (X2,Y2,B2,H2,$9600+B1*H1);
67   Ch:= ReadKey;
68   RestSprite (X1,Y1,B1,H1,$9600);
69   Ch:= ReadKey;
70 End.
```

Wenn Sie Ihre Menüs auslüften wollen, öffnen Sie doch ein „window“. Windemo.pas können Sie durchaus als Fenstergriff verwenden.

gramm SUNSHADE.PAS zeichnet einzelne Phasen eines Bewegungsablaufes, legt sie im ungenutzten Bildspeicher ab und holt sie dann mit steigendem Tempo auf den Bildschirm zurück. So könnte man Mickey Mouse' Sonnenschirm zum Leben erwekken. Da fährt auch schon ein lauer Nachtwind unter das Schirmchen und läßt es sanft rotierend entschweben. Mickey hätte ihn besser nicht am Strand vergessen sollen.

Ein abendfüllender Spielfilm wird es wohl nicht werden – stehen doch für die ganze Handlung nur 27136 Byte zur Verfügung. Das Programm verwendet zur Animation der Handlung 26 Bildchen zu je 128 × 64 Pixel, also ein KByte je Bild. Dies kleine Beispiel zeigt eine Möglichkeit, wie periodische Vorgänge, deren einzelne Phasen zuviel Rechenzeit kosten würden, mittels Einzelbildsequenz in eine halbwegs fließende Darstellung auf den Bildschirm zu bringen sind.

Literatur

[1] Jürgen Petsch, Schnellfärberei, c't 8/89, S. 176, c't 9/89, S. 214

Nachdruck aus c't – Magazin für Computertechnik, Heft 4/1990

Turbo-Pascal-Tip

Heller Hintergrund im Textmodus

MS-DOS vergibt bei der Nutzung einer EGA-/VGA-Karte im Textmodus für jedes Zeichen auf dem Bildschirm ein spezielles Farbattribut, das heißt einen Wert zwischen 0 und 255, der angibt, welche Vorder- und Hintergrundfarben das Zeichen hat und ob es blinkt. Normalerweise liegen nur die dunklen Farben (0 bis 7) als Hintergrundfarben vor. Der Bereich ab 128 kennzeichnet dabei die gleichen Attribute wie der Bereich bis 127, allerdings mit dem Untcrschied, daß die Vordergrundfarben blinken. Mit der im Bild 1 angegebenen Prozedur LightBack kann die Bedeutung der Farbattribute ab 128 umgeschaltet werden. Nach Aufruf der Prozedur werden die Vordergrundfarben normal, also nicht blinkend, und die Hintergrundfarben hell leuchtend dargestellt. Und so wird aus blinkend schwarz auf grau (240) mit LightBack schwarz auf weiß. Die Prozedur BlinkBack schaltet wieder auf die übliche Darstellung um.

Die beiden Prozeduren eignen sich besonders zur Hervorhebung von Leuchtbalken (beispielsweise zur Auswahl in Menüs) oder für Texteditoren, bei denen es auf einen hellen Hintergrund ankommt.

Bernd Beyreuther

```
  Programm zeigt, wie mit EGA/VGA Karte
  im Textmode von blinkenden
  Farbattributen auf helle Hintergrund-
  farben umgeschaltet wird.
  Bernd Beyreuther , 19.März 1990,
  getestet auf IBM AT(VGA mit 16 Farben),
  DOS 3.3,
  turboPASCAL 4.0/5.0
--------------------------------------}
program BLINK_LIGHT;
uses dos,crt;
var regs:registers;
{-------------------------------------}
 procedure lightBACK;
 begin
  regs.ah:=16;
  regs.al:=3;
  regs.bl:=0;
  intr(16,regs)
 end;
{-------------------------------------}
 procedure blinkBACK;
 begin
  regs.ah:=16;
  regs.al:=3;
  regs.bl:=1;
  intr(16,regs)
 end;
{-------------------------------------}
 procedure say(s:string);
 begin
  writeln(s);
  readln
 end;
{-------------------------------------}
begin
 clrscr;
 blinkback;       { ATTR > 127 blinkend }
 textattr:=112;
 say('NORMALES SCWARZ AUF GRAU');
 textattr:=112+128;
 say('BLINKENDES SCHWARZ AUF GRAU');
 lightBACK;       { ATTR > 127 intensiv }
 say('SCHARZ AUF WEISS');
 blinkback;
 say('UND WIEDER NORMAL BLINKEND')
end.
{-------------------------------------}
```

Wegbereiter der Informatik

HOWARD HATHAWAY AIKEN

*** 1900 Hoboken (New Jersey)**
† 1973 St. Louis (Missouri)

H. H. Aiken hat zunächst Elektrotechnik studiert und in diesem Fachgebiet an der Universität von Wisconsin 1923 ein Examen abgelegt. Schon während seiner Studienzeit war er seit 1919 in einer Elektro-Firma in Madison als Ingenieur für Energieversorgung beruflich tätig; nach seinem Examen befaßte er sich dort mit Konstruktionsproblemen von Starkstromgeneratoren. Mit 31 Jahren begann Aiken, ein zweites Mal zu studieren, diesmal belegte er an der Universität Chicago die Fächer Mathematik und Physik. Im Jahre 1932 wechselte er an die Harvard-Universität Cambridge über, legte hier 1937 sein zweites Examen ab („Master of Science") und promovierte 1939. Zwei Jahre später erhielt er an dieser Universität eine Professur für angewandte Mathematik, die er 22 Jahre innehatte. Ab 1961 bekleidete er dann eine Professur für Informatik an der Universität Miami in Florida.

Während des 2. Weltkrieges erhielt Aiken von der Harvard-Universität den Auftrag zum Bau eines Digitalrechners. Auf Grund seiner 2fachen Ausbildung als Elektrotechniker und Mathematiker war er offenbar der geeignete Wissenschaftler, einen solchen Auftrag in die Tat umzusetzen. Der Babbage-Biograph Anthony Hymann berichtet (1987), daß Aiken von Babbages Maschinen besonders begeistert gewesen ist und auch dadurch mit dazu inspiriert wurde, einen solchen Auftrag von sich aus vorzuschlagen. Von Zuses bereits 1941 fertiggestelltem programmgesteuerten Rechenautomaten Z3 kann Aiken angesichts der damaligen Kriegssituation kaum etwas gewußt haben, denn Zuses Arbeiten blieben damals außerhalb Deutschlands jahrelang unbekannt. Während Zuse in Anlehnung an Leibnizsche Ideen in seiner Maschine bereits das Dualsystem und die Gleitkommaarithmetik realisiert hatte, verwendete Aiken in seinem Automaten bei der Ausführung der Rechenoperationen eine dekadische Zahlendarstellung. Aikens Maschine, die unter seiner Leitung von der Firma IBM gebaut wurde, war 1944 vollendet und erhielt die Bezeichnung MARK I (mancherorts auch ASCC = Automatic Sequence Controlled Calculator genannt). MARK I ist der erste in Amerika konstruierte, teilweise programmgesteuerte elektromechanische Digitalrechner; seine Operationszeit betrug für eine Addition 0,3 Sekunden, für eine Multiplikation 6 und für eine Division 15 Sekunden. Die Dateneingabe erfolgte über Lochstreifen.

MARK I ist 15 Jahre lang genutzt und noch verbessert worden. So wurden 1947 der Nachfolgerechner MARK II fertiggestellt und später die Automaten MARK III und MARK IV, die dann vollständig mit Elektronenröhren ausgerüstet waren (MARK I und MARK II waren Relaisrechner).

Von Aiken stammt auch ein spezieller Binär-Dezimal-Code, bei dem jede Dezimalziffer als Tetrade verschlüsselt wird (*Aiken-Code*); dabei handelt es sich um einen Komplementärcode, der günstige Eigenschaften bei der technischen Realisierung der Dezimalarithmetik aufweist.

Howard Aiken erhielt für seine computertechnischen Entwicklungsarbeiten sowie für seine theoretischen Beiträge zu Fragen der Codierung und Programmierung etc. zahlreiche Auszeichnungen, zum Beispiel von der französischen (1949) und von der belgischen (1951) Akademie. 1952 wurde er zum Ritter der Ehrenlegion ernannt und war Ehrendoktor der Technischen Hochschule Darmstadt und der Universität Wayne.

Dr. Klaus Biener

Computer-Club

BICLAN – Ein lokales Netz für Lehrkabinette

Dr. Christian Löber, Dresden

BICLAN ist ein lokales Netz für den Bildungscomputer A 5105 (BIC). Der vorliegende Bericht gibt einen Überblick über Eigenschaften und Anwendungsmöglichkeiten von BICLAN.

Einsatzgebiet

Hauptanwendung für BICLAN ist die Vernetzung des Bildungscomputers A 5105 in Lehrkabinetten. Damit sind die Leistungsmerkmale wesentlich bestimmt durch die schulspezifischen Einsatzbedingungen. Die typische Ausstattung eines Lehrkabinettes besteht aus einem Lehrerrechner (L_BIC) und 7 Schülerrechnern (S_BIC).

BICLAN ist gekennzeichnet durch eine kostengünstige und problemlos zu installierende Netzhardware. Die Ankopplung der zu vernetzenden Rechner erfolgt durch einfaches Einketten in das als Übertragungsmedium dienende Koaxialkabel mit den Mitteln der passiven Antennenanschlußtechnik. BICLAN ist somit ein typisches Beispiel für ein Low-cost-LAN. Es ist unter beiden Betriebssystemen des A 5105 – RBASIC und SCPX – lauffähig.

Hardwarekonzept

BICLAN basiert auf einer modifizierten Version des lokalen Netzes LOTUNET /1/. Die die LAN-Hardware repräsentierende physische Schicht PH enthält folgende Komponenten:

– *Die Medienanschlußeinheit MAU.* Sie besteht beim BICLAN lediglich aus einer simplen T-Anschlußdose der Antennentechnik, über die ein zu vernetzender A 5105 in das Koaxialkabel des LANs eingekettet wird. Dabei wird als Netztopologie eine Linienstruktur verwendet (Bild 1).

– *Die Netz-Interface-Einheit NIU.* Sie realisiert logisch und physisch die Kopplung mit dem Übertragungsmedium und steuert die Datenkommunikation. Hauptkomponente ist eine im synchronen SDLC-Verfahren arbeitende SIO U 856, die die bitorientierte Übertragung der Daten organisiert.

Als Übertragungsprinzip benutzt BICLAN das stochastische CSMA/CD-Zugriffsverfahren. Die Netzteilnehmer verfolgen den Datenverkehr auf dem Netz ständig mit. Beim Empfang erkennt ein Teilnehmerrechner die für ihn bestimmte Nachricht anhand der Stationsadresse. Beim Senden kann die kritische Situation des gleichzeitigen Sendebeginns mehrerer Rechner entstehen. Sie wird dadurch aufgelöst, daß die Sender nach Sendeabbruch erst nach einer für die einzelnen Rechner unterschiedlichen Wartezeit erneut senden. Dies ist eine der Grundeigenschaften des CSMA/CD-Verfahrens, allerdings mit dem Nachteil einer nicht determinierten Übertragungszeit. Für die Lösung der Aufgaben von BICLAN ist dies jedoch von untergeordneter Bedeutung.

Das Schichtenmodell von BICLAN

Die Datenverbindungsschicht DL stellt Dienstprimitiven und dazugehörige Statusinformationen für Nutzer dieser Schicht zur Verfügung. Die Primitiven beinhalten Dienste zur Netzinitialisierung sowie zum Senden und Empfangen von Telegrammen. Die Übertragung basiert auf dem Prinzip des Telegramm-Handshakes, das heißt, nach jeder gesendeten Nachricht wartet der Sender auf die Antwort des adressierten Empfängers. Die für Netzübertragungen kritischen Situationen von Verlust oder Dopplung von Telegrammen werden über einen Quittungs- und Flußsteuerungsmechanismus automatisch erkannt und behoben. Auf die Schichten PH und DL ist die datenverarbeitende Applikationsschicht APP für die beiden Betriebssysteme RBASIC und SCPX 5105 augesetzt. Diese wird durch die Dienste von BICLAN repräsentiert. Damit realisiert BICLAN eine 3-Schichten-Architektur nach dem OSI-Referenzmodell, wobei die Schichten 3 bis 6 leer sind (Tafel 1).

Softwarekomponenten

Das sowohl residente wie transiente Komponenten enthaltende Programmsystem von BICLAN basiert auf einem schichtenorientierten Schnittstellenkonzept. Die residenten Komponenten werden beim Kaltstart automatisch, die transienten Komponenten im Rahmen eines erweiterten Boot-Prozesses nachgeladen. BICLAN basiert auf folgenden Grundforderungen:

– keine direkte Kommunikation zwischen Schülerrechnern
– keine Vorkenntnisse netzspezifischer Kommandos beim Schüler
– Einleitung und Kontrolle aller das Netz betreffenden Aktivitäten nur durch den Lehrer (Masterfunktion)
– Lauffähigkeit unter RBASIC und SCPX 5105.

Wegen dieser Forderungen ist die in den Schülerrechnern und im Lehrerrechner laufende Netzsoftware unterschiedlich.

Kommandoregime

BICLAN basiert auf der Nutzung der dominanten Master-Slave-Relation, das heißt, jede Informationsübertragung wird vom Lehrer als Auftraggeber eingeleitet und vom ausgewählten Schülerrechner unmittelbar beantwortet. Die Einleitung und die Überwachung der Netzaktivitäten wird durch ein Bedienermenü für beide Betriebssysteme unterstützt. Durch die kursorgesteuerte Bedienerführung sind alle Dienste von BICLAN auf einfache und übersichtliche Weise und ohne eine langwierige Lernphase nutzbar. Der Beginn einer Kommunikationsphase ist durch den Übergang des ausgewählten Schülerrechners in den Slavestatus gekennzeichnet, das heißt, ab sofort hat nur noch der Lehrer Zugriff auf die Ressourcen des Schülerrechners. Nach Abarbeitung des Netzkommandos wird der Slavestatus des Schülerrechners automatisch wieder aufgehoben.

Status und Betriebsweise eines Schülerrechners werden je nach Netzkommando über unterschiedliche Schnittstellen des netzfähigen Betriebssystems beeinflußt:

– durch direkte und sofortige Auswertung des über Interrupt empfangenen Netztelegramms, beispielsweise bei der Aufforderung an den Schülerrechner, den Slavestatus einzunehmen
– über die zeichenorientierte BIOS-Schnittstelle bei der Stringverarbeitung; dies ist beim Druckerdienst der Fall.
– über die sektororientierte BIOS-Schnittstelle bei Netzkommandos mit Dateiarbeit, beispielseise beim Filetransfer.

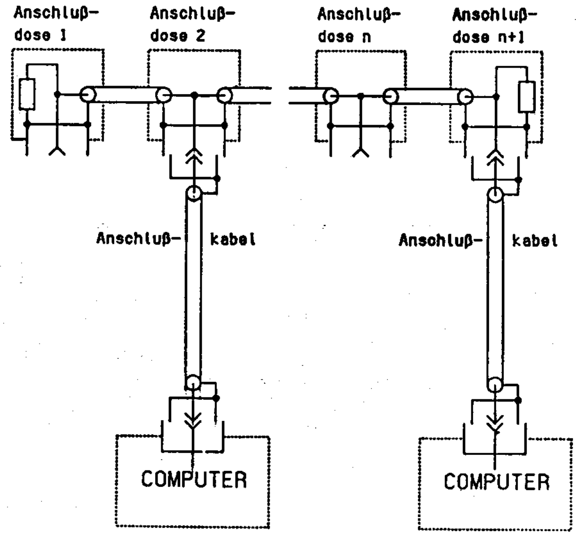


Bild 1 Netztopologie des lokalen Netzes BICLAN

Tafel 1 Das Schichtenmodell des lokalen Netzes BICLAN

Schicht			Nr.
Applikationsschicht	APP	Dienste von BICLAN	7
		leer	6
		leer	5
		leer	4
		leer	3
datenverarbeitende Schicht DL	LLC	logische Verbindungssteuerung	
	MAC	Kanalzugriffssteuerung	2
physische Schicht PH	PLS	bitorientierte Übertragung	1
	PMA	Medienanschluß	

Dienste von BICLAN

Hauptdienst von BICLAN ist der Filetransfer. Er ermöglicht die Übertragung von sequentiellen oder Direktzugriffsdateien beliebiger Länge. Dies können Programm-, Text- oder Datenfiles sein. Bei hardwareseitigem Vollausbau verfügt der Lehrerrechner über folgende Laufwerkskonfiguration:

A physisches Laufwerk in der Diskettenstation DSE
B, C physische Laufwerke in eventuell vorhandener Diskettenbeistellung
D, E RAM-Floppies
F...P LAN-Laufwerke von BICLAN (gleich physische Laufwerke A der Schülerrechner 1...11).

Files können also zwischen allen 16 logischen Laufwerken transferiert werden, vorausgesetzt, die physischen Ressourcen hierfür sind vorhanden.

Bei normal ausgestatteten Computerkabinetten hat der Lehrer Zugriff auf sein physisches Laufwerk A sowie auf die 7 LAN-Laufwerke F...L, korrespondierend mit den physischen Laufwerken A der Schülerrechner S_BIC1 ... S_BIC7 mit den Stationsadressen 1...7. Die Nettoübertragungsrate beträgt beim Filetransfer zirka 1 KByte pro Sekunde.

Literatur

/1/ Weidhase, F.: Hardwarekomponenten lokaler Netze. Technische Universität Dresden, Informatik-Zentrum, 1988
/2/ VEB Robotron-Meßelektronik Dresden: BICLAN – Lokales Netz für den Bildungscomputer A 5105, Bedienungsanleitung

Commodore Amiga 3000

Zur Sicob-Messe in Paris und in New York präsentierte Commodore Mitte April das neue Flaggschiff seiner Amiga-Reihe, das die Firma vornehmlich als Workstation im kreativen Bereich sieht. Aufgaben vom Desktop Publishing über die digitale Bildbearbeitung bis hin zur Produktion interaktiver Verkaufsinformationssysteme sollen mit dieser Maschine bewältigt werden. Commodore zielt also – wie neuerdings auch Apple mit dem Mac IIfx; s. S. 35 – vor allem auf den als wachstumsträchtig eingeschätzten professionellen „Multimedia-Markt". Multimedia ist die Verknüpfung einzelner Bausteine wie Realbild-Video, Computergrafik und -text, Standfotos, Stereomusik und -geräusche zu kompletten Präsentationen oder Lern- und Informationssystemen. Das soll der Amiga 3000 noch einfacher und kostengünstiger als das bisherige Spitzenmodell Amiga 2500/30 leisten – und zwar im Einsatz auf dem Schreibtisch, wie mit einem Standard-PC.

Multimedia-Software

Konkrete Multimedia-Anwendungen sind etwa Desktop-Präsentationen, deren Anspruch über farbige Balkendiagramme deutlich hinausgeht, die aber nicht die Produktionskosten von Videoclips haben sollen. Andere typische Beispiele dafür sind Informationssysteme in Flughäfen oder Kaufhäusern oder interaktive Lernprogramme, die Auge und Ohr gleichermaßen ansprechen. Zur einfachen Anwendung der Amiga-3000-Möglichkeiten wird ein Multimedia-Autorensystem unter der Bezeichnung *AmigaVision* gleich mitgeliefert. Programmierkenntnisse sollen nicht erforderlich sein. Mit Hilfe von Symbolen am Bildschirm werden sowohl die diversen Multimediaquellen – von der TV-Kamera über den Bildplattenspieler, vom Tonbandgerät bis hin zur eingebauten Festplatte des Amiga – aufgerufen, als auch die gewünschten Informationen ausgetauscht und in ihrem Ablauf festgelegt.

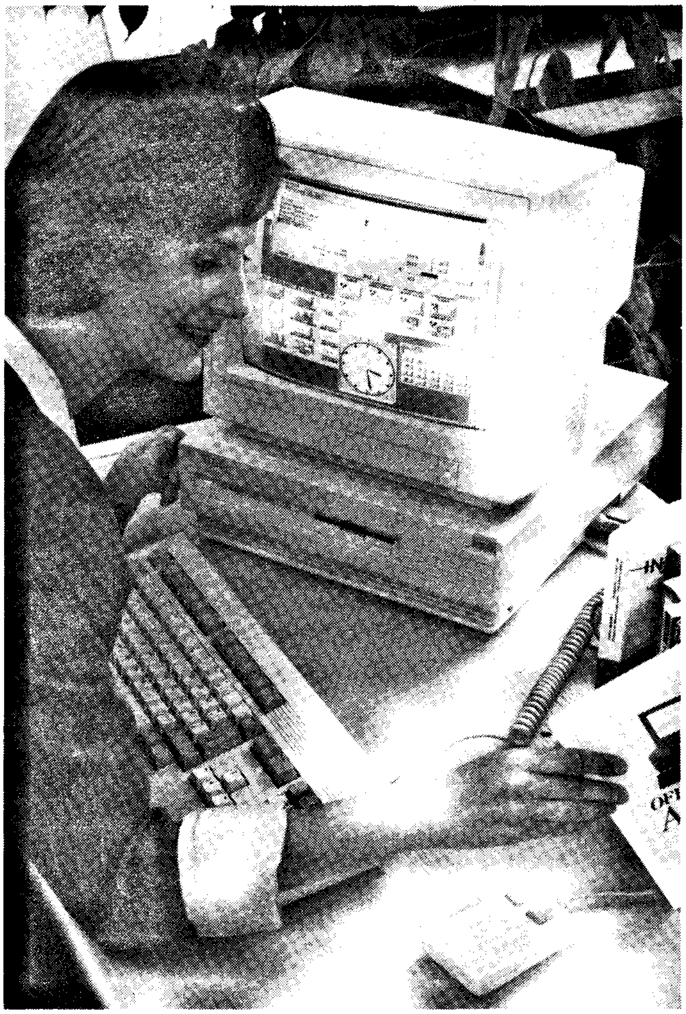
Drei Modelle

Allen Amiga-3000-Versionen ist neben der serienmäßigen Festplatte von mindestens 40 MByte gemeinsam, daß jetzt der Motorola-Chip 68030 direkt auf der Mutterplatine sitzt und nicht mehr auf einer Zusatzkarte wie beim Amiga 2500. Diese Zusatzkarte wird allerdings als Ausbaumöglichkeit für den Amiga 2000 weiterhin angeboten – genau wie der Amiga 2500/30 selbst. Ebenso ist eine Brücke zu den industriestandardkompatiblen PCs im Angebot. Unter Verwendung von Zusatz-Einbaukarten ist der Amiga 3000 in der Lage, mit entsprechenden Programmen zu arbeiten und MS-DOS-Daten zu lesen und zu schreiben. Die Plazierung des Hauptprozessors auf der Mutterplatine und weiter verbesserte acht Spezialchips, unter anderem für Bildaufbau, Datenmanagement, Tonausgabe, machen die Maschine gemeinsam mit dem überarbeiteten Multitasking-Betriebssystem – jetzt Amiga-DOS 2.0 – leistungsfähiger als den Amiga 2500/30.

Der *Amiga 3000-16-40* ist mit 16 MHz getaktet, besitzt neben dem Hauptprozessor noch den mathematischen Koprozessor 68881, 1 MByte Chip-RAM (Arbeitsspeicher für den Hauptprozessor und die Koprozessoren) sowie 1 MByte Fast-RAM (Arbeitsspeicher, der allein dem Hauptprozessor zur Verfügung steht). Beide Arbeitsspeicher können mit 1-MBit-Chips auf insgesamt 6 MByte aufgerüstet werden, mit den jetzt auf den Markt kommenden 4-MBit-Chips sogar auf 18 MByte.
Die beiden mit 25 MHz getakteten Amiga 3000 tragen die Bezeichnung *3000-25* und zusätzlich die Größe der Festplatten von 40 und 100 MByte, die beide mit einer mittleren Zugriffszeit von unter 19 Millisekunden arbeiten. Der mathematische Koprozessor dieser beiden Versionen ist der MC68882.
Alle Versionen sind mit dem üblichen 3,5-Zoll-Diskettenlaufwerk (880 KByte) ausgestattet.
In einem kleineren Gehäuse als beim Amiga 2000 sind für Erweiterungen noch acht Steckplätze frei, die aber nicht alle gleichzeitig belegt werden können. Zwei der vier Amiga-Steckplätze (Zorro III, abwärtskompatibel zu Zorro II) können auch als PC/AT-Steckplätze genutzt werden. Zusätzlich stehen ein Video-Steckplatz (z. B. für Genlocks zur Verbindung mit TV-Technik) und ein Erweiterungssteckplatz für Koprozessor-Karten und Arbeitsspeicher bis in den Gigabyte-Bereich hinein zur Verfügung. Ein gegenüber dem Amiga-Standard verbesserter Videoadapter (VDE – Video Display Enhancer) und ein SCSI-Controller sowohl für die interne Festplatte wie für den externen SCSI-Anschluß sind bereits auf der Mutterplatine installiert und benötigen keine der vorhandenen Steckplätze. An den neuen VDE-Ausgang kann ein handelsüblicher Multiscan-Monitor ohne Zusatzkarte angeschlossen werden. Ein Ausgang für den Commodore-Monitor 1084 ist ebenfalls vorhanden.

Verbessertes Betriebssystem

Das neue Amiga-DOS mit der Workbench 2.0 ist ein Betriebssystem, das mit seiner leichten Bedienbarkeit, Multitasking-Fähigkeit und integrierten Fenstertechnik professionellen Ansprüchen gerecht wird und sich gut als Umgebung für hochwertige Anwendungssoftware eignet. Gegenüber der bisherigen Workbench 1.3 wurde nicht nur der optische Eindruck der Benutzeroberfläche verbessert, sondern auch die Geschwindigkeit des Betriebssystems durch komplette Programmierung in der Computer-Hochsprache C erhöht.
Zusätzliche Funktionen im Betriebssystem sind z. B. das Sortieren von Inhaltsverzeichnissen (Directories), die einfache Einrichtung einer Startdatei (um direkt beim Einschalten des Amiga in eine bestimmte Anwendungssoftware hineinzugehen) und die Software-Kommunikationsschnittstelle A-Rexx, um den Datentransfer zwischen einzelnen Programmen zu vereinfachen. *MP-We*



Die Hardware im Überblick

Hauptprozessor

Motorola 68030, 16 MHz, mit mathematischem Koprozessor 68881 oder 68030, 25 MHz, mit mathematischem Koprozessor 68882

Arbeitsspeicher

1 MByte Chip-RAM, erweiterbar auf 2 MByte auf der Hauptplatine
1 MByte Fast-RAM, erweiterbar auf 4 (16) MByte auf der Hauptplatine

Steckplätze

4 Zorro-III-Slots, davon 2 in Linie mit PC/AT-kompatiblen Slots; 1 Video-Slot für Genlocks usw.; 1 Slot für CPU/Fast-Speicher

Anschlüsse

Amiga-Video, 23 Pins (15,625 kHz); VGA-Typ-Video, 15 Pins (31,25 kHz), für Non-Interlace; Stereo-Audio rechts und links; externer SCSI-Anschluß (25 Pins); interner SCSI-Anschluß (50 Pins) für 40- oder 100-MByte-Harddisk; externes Plattenlaufwerk (Floppy oder Harddisk); seriell, parallel

Programmieren mit FORTH

von G.-U. Vack, Verlag Technik, Berlin 1990, 336 S., 85 Bilder, 17 Tafeln, 35,– DM, Best.-Nr. 5539181

Da es sich um das erste größere Forth-Buch aus einem DDR-Verlag handelt, ist der Autor erfreulicherweise bemüht, das Spektrum der angesprochenen Leserschaft möglichst breit zu halten. Nach einem einführenden Kapitel (Programmiersprachen und Betriebssysteme, 14 Seiten) versucht er im zweiten Kapitel (Grundlagen der FORTH-Programmierung; 32 Seiten) auch für Neulinge eine gewisse Einstiegsmöglichkeit zu bieten. Funktionieren könnte das in den Fällen, wo deutliches Interesse am Forth-Konzept mit guten Vorkenntnissen in der Rechentechnik zusammenfällt. Gerade der Einsteiger wird hier aber leider durch Ungenauigkeiten (z. B. in Bild 2.4) irritiert. Oberflächeneigenschaften von Forth werden in einer Anzahl Definitionen und Merksätzen im Text beschrieben. Für deren ins einzelne gehende Begründung aus dem Forth-Konzept heraus ist leider zu wenig Platz. Das ist bedauerlich, weil eine der Stärken von Forth gerade darin besteht, daß dem Benutzer der organische Zusammenhang von Oberflächeneigenschaften und internem Aufbau prinzipiell zugänglich ist und hier eine wesentliche Quelle zum Verständnis des Konzepts liegt.
Kapitel drei (Konzepte der Implementierung von FORTH, 27 Seiten) geht auf die Komponenten Wörterbuch, Fadencode, Stacks und Massenspeicher ein. Eine Vielzahl von Bildern unterstützt den gewissenhaften Leser bei der Aufnahme von Wissen aus dem wenig redundanten Text.
Kapitel vier (Glossarium von FIG-FORTH und von FORTH-83, 91 Seiten) und Kapitel fünf (Weitere Festlegungen von FORTH-83, 18 Seiten) enthalten den Befehlssatz in thematischer Sortierung, wieder unterstützt durch Bilder und Definitionsbeispiele. Die gleichzeitige Behandlung der zwei verschiedenen Standards ist ein Zugeständnis an den noch nicht überall vollzogenen Übergang zum moderneren Standard FORTH 83. Günstig ist das konzentrierte Eingehen auf den 83er Standard und die Zusammenstellung der Unterschiede beider Standards im Kapitel fünf.
Kapitel sechs (Kompendium der FORTH-Programmierung, 61 Seiten) zeigt die Codierungen von besonders häufig gebrauchten Spracherweiterungen. Das wird in weiten Teilen gern als Vorlage zum Abtippen benutzt werden.
Kapitel sieben (Ausgewählte Komponenten einer FORTH-Programmierumgebung, 21 Seiten) behandelt u. a. Editor, Assembler und Debugger, für die jeweils vollständige Beispiele abgedruckt sind. Angenehm sind für Einsteiger sicherlich die praktischen Tips, z. B. zum Booteditor. Überhaupt enthalten die Kapitel sechs und sieben eine Fülle von Beispielen, die im Sinne von Prototypen vom Leser in vielfältiger Weise weiterentwickelt werden können.
Kapitel acht (Aspekte der Softwareentwicklung und Softwaregestaltung mit FORTH, 23 Seiten) bringt Richtlinien in knapper Form und großer Zahl. Der Programmierer kann hier sein Heil nicht im Streben nach der möglichst vollständigen formalen Befolgung aller Einzelhinweise suchen. Was in der Programmierpraxis wo sinnvoll zu verwenden ist, wird sich immer aus der Berücksichtigung der konkreten Umstände ergeben. Fast die Hälfte des Kapitels präsentiert als vollständiges Beispiel eine GKS-Grafikeinbindung für den Rechner A 7100. Zwei weitere Kapitel befassen sich noch einmal kurz mit Implementierungsfragen und einem Ausblick auf weitere Forth-Entwicklungen in Hardware und Software.
Ein 16seitiger Anhang bietet Worterklärungen, Übersetzungen englischer Begriffe und ein Register von Forth-Schlüsselwörtern.
Abgerundet wird das Buch mit einem jeweils vierseitigen Literatur- und Sachwörterverzeichnis.
Dem Autor ist es gelungen, eine nützliche Wissenssammlung zum Nachschlagen für Forth-Programmierer anzulegen, gleichzeitig aber auch Vertreter anderer Programmiersprachen zu einem „Blick über den Zaun" einzuladen. Ein Buch zum Durchlesen ist das allerdings nicht. Hier konkurrieren das Streben nach Vollständigkeit mit dem Platzmangel von 336 Seiten. Gemessen am Inhalt hätte das Buch anstatt der Nadeldruckervorlage einen ordentlichen Satz und ein früheres Erscheinen verdient. Vereinzelte kleinere Unschönheiten (z. B. im Abschnitt „Rekursive Programmierung") sollten sich in einer nächsten Auflage beheben lassen.
Fazit: Für den aktiven Forth-Programmierer findet sich genügend interessanter Stoff zur Information und zur Auseinandersetzung. Ausgesprochene Forth-Neulinge werden allerdings nicht ohne zusätzliche – speziell einführende – Literatur auskommen.

H. Pfüller

Desktop-Knigge

von P. Luidl, te-wi-Verlag, München 1988, 196 S., ISBN 3-921803-85-3, 79,– DM

Ein Buch, mit dem der Autor laut Vorwort Desktop-Publishern das traditionsreiche und gewachsene Wissen von Schriftfachleuten darstellen will, damit sie mit ihren DTP-Systemen mehr machen können, als nur Texte elektronisch schreiben und verwalten – nämlich auch Schrift setzen und Seiten gestalten. Das gute Benehmen im Umgang mit der Schrift also.
Damit beschränkt sich der Autor keinesfalls nur auf das hierfür notwendige Wissen eines Setzers, sondern berücksichtigt auch das eines Typografen und Gestalters. Der Eindruck, den das Buch von der didaktischen Aufbereitung her hinterläßt, ist zwiespältig: einerseits die Vermittlung von notwendigem „Einsteigerwissen" (Formate, Falzarten, Satzmaterial zum Beispiel), andererseits die sehr knapp gehaltene Darstellung, die mehr einem Nachschlagewerk für Profis gerecht wird als einem Einsteigerbuch. Mit dem erläuternden Text zum Schwerpunkt wird vielleicht doch etwas zu sparsam umgegangen. Wie dem auch sei: Das Buch bietet sicher jeder Gruppe etwas, da das Spektrum von den Grundbegriffen der Typografie über die Elemente ästhetischer Schriftgestaltung, die Seitengestaltung von Zeitschriften und Büchern bis zur Gestaltung von Pressemitteilungen und Schachteln reicht. So wird es sowohl der DTP-Neuling als auch der routinierte Publisher mit Gewinn nutzen können.

H. Weiß

Einführung in Objektorientierte Programmierung mit Turbo Pascal 5.5

von J. Mittendorfer, Addison-Wesley (Deutschland) GmbH, Bonn 1990, 330 S., mit Programmdiskette, 68,– DM, ISBN: 3-89319-248-4

Um es vorwegzunehmen: Das Buch stellt eine gelungene Ergänzung zu den Borland-Handbüchern dar und kann allen Turbo-Pascal Anhängern empfohlen werden, die den Umstieg zur objektorientierten Programmierung in Angriff nehmen wollen. Dieses Angebot gilt nicht nur für den fortgeschrittenen Programmierer; ausreichend sind grundlegende Pascal-Kenntnisse. So geht es dann auch nicht vordergründig um die Darstellung neuer Syntax und Befehle (nur vier neue Schlüsselwörter würden allenfalls ein Faltblatt füllen), sondern um die Entwicklung einer völlig anderen Denkweise, einer neuen Sicht auf Daten- und Programmstrukturen. Nicht mehr passive Daten werden durch Funktionen manipuliert, vielmehr enthalten sie nun selbst Methoden zu ihrer Veränderung. Hat man sich dieses grundsätzliche Prinzip erst einmal zu eigen gemacht, so wird die zukünftige Arbeit entscheidende Erleichterungen erfahren. Besondere Impulse erhofft man sich für die zukünftige Gestaltung des Nutzerdialogs, vor allem im Hinblick der Entkopplung vom eigentlichen Problem und der Wiederverwendung einmal programmierter und getesteter Module.
Das Buch wird dem Anspruch einer Einführung gerecht. In zwei Kapiteln wird Grundwissen vermittelt und die neue Begriffswelt hergeleitet. Dazu holt der Autor weit aus, bringt Historisches, Vergleiche und Zitate. Gute Beobachtungsgabe zeigt er bei der Darstellung gängiger Programmierpraxis, die dem Leser die objektorientierte Programmierung als Ausweg zwangsläufig erscheinen läßt.
Die restlichen 5 Kapitel sind der Herleitung von Beispielen und einem kleinen Ausblick in die Zukunft gewidmet. Alle Beispiele sind so gewählt, daß sie die Vorteile objektorientierter Programmierung gut zur Geltung bringen und für den Nutzer auch praktisch verwendbar sind. Dazu gehören ein Zeitzähler, Maus- und Cursorfunktionen für die Dialogbehandlung, so daß mancher diese Techniken, um die er ob ihrer Kompliziertheit bisher einen Bogen gemacht hat, jetzt einmal selbst versuchen wird. Im weiteren geht es um dynamische Objekte (Listen, Stacks, Streams), für die sich mittels objektorientierter Programmierung alle wichtigen Funktionen vordefinieren lassen, die dann auf Objekte unterschiedlicher Struktur anwendbar sind und viel Programmieraufwand sparen. Die schwer vorstellbare dynamische Methodenzuordnung ist zudem mit Abbildungen illustriert. Da Fehler trotzdem nicht ausbleiben werden, wird in kurzer Form auch die Handhabung des Turbo-Debuggers 1.5 abgehandelt.
Nicht notwendig wäre es gewesen, daß Paradebeispiel der graphischen Objekte mit aufzunehmen, das im Turbo-Handbuch ausführlich dargestellt ist. Gut dagegen ist die Idee, die gleiche Typographie wie in den Handbüchern zu verwenden (Schlüsselwörter fett, Namen kursiv). Turbo-Pascal schafft sich eine andere Begriffswelt als die objektorientierten Vorbilder Smalltalk und C++. Im Glossar wird darauf hingewiesen. Außerdem befindet sich im Anhang eine Übersicht über alle Funktionen der Units, die sich auf der Beispieldiskette befinden und die im Verlaufe des Buches hergeleitet wurden. So wird denn die objektorientierte Programmierung hoffentlich nicht nur die Arbeit erleichtern, sondern auch Spaß machen. Zitat von D. Ingalls auf Seite 28: „Statt eines Bit-zermalmenden Prozessors, der Datenstrukturen zerfetzt und vergewaltigt, haben wir in einem objektorientierten System eine Welt von Objekten, die sich höflich gegenseitig darum bitten, diverse Wünsche auszuführen". Na, bitte.

U. Schulze

Datenbank-Konsultationszentrum

Daten sind eine wertvolle Ressource. Datenbestände beinhalten die Informationen, die das Funktionieren Ihrer Einrichtung sichern sollen. Die passende Struktur und die zweckmäßige Verwaltung dieser Datenbestände sind die entscheidenden Grundlagen für die rationelle Bewältigung ihrer Aufgaben. Dafür die angepaßten und kostengünstigen Entscheidungen schnell und sicher zu treffen ist besonders wichtig. Wir wollen Ihnen dabei helfen, indem wir Ihnen unser Fachwissen zur Verfügung stellen. Unsere Leistungen beinhalten: Beratung bei der Auswahl von Datenverwaltungssystemen, Beratung bei Entwurfsfragen, Unterstützung beim konzeptionellen logischen und physischen Entwurf, Beratung beim Tuning von Datenbank-Leistungen.

VEB Leitzentrum für Anwendungsforschung, DB-Konsultationszentrum, Bereich F2, Jacques-Duclos-Straße 46–52, Berlin, 1156; Tel. 37 80 35 10 (Herr Kluge) oder 37 80 35 00 (Herr Dr. Lange)

Dr. Warnow

Riesenlaptop mit Symbolcharakter: Die Bedeutung der tragbaren PC-Zwerge ist inzwischen so riesengroß, daß sie nicht mehr übersehen werden können. Toshiba gelang es mit diesem überdimensionalen, funktionsfähigen Modell des T 5200 übrigens, ins Guinness-Buch der Rekorde zu kommen.

Im März dieses Jahres war es wieder soweit: Die weltgrößte Computerfachmesse gab über 4000 Ausstellern Gelegenheit, ihre Neuheiten zu präsentieren und den über 560 000 Besuchern die Chance, sich einen Überblick zu verschaffen oder für sich das „Richtige" zu finden. Besonders den knapp 25 000 Besuchern aus der DDR, von denen die meisten erstmals eine CeBIT kennenlernen konnten, dürfte ob der Ausmaße bald der Kopf geschwirrt haben. Wer auf der Suche nach sensationellen Neuheiten war, sah sich nach allgemeinem Urteil enttäuscht: Es gab keine. Für die meisten Besucher allerdings sicher nebensächlich; die Möglichkeit des konzentrierten Überblicks schlägt eindeutig positiv zu Buche. Zumindest für den Fachmann. Dem „Otto-Normal-PC-Anwender" oder künftigen PC-Nutzer – insbesondere aus der DDR – wären überschaubare Veranstaltungen wie die COM in Erfurt wohl eher zu empfehlen.
Mit der folgenden Zusammenstellung der Neuheiten erhalten Sie einen Überblick über das, was derzeit international „Up to date" ist; ob es für Sie wirklich notwendig ist, dürfte eine andere Sache sein. Denn für die meisten Anwender reichte es auch schon „eine Nummer kleiner".

Hardware

Das Besucherinteresse richtete sich verständlicherweise vorrangig auf die Welt der Personalcomputer; Mainframes oder Minis waren auf der CeBIT von untergeordneter Anziehungskraft und auch kaum ausgestellt. Etwas anders war es bei der Klasse der leistungsfähigen Workstations. Hier standen mehrere Hersteller – darunter auch Außenseiter für spezielle Anwendungen, vor allem im CAD-Bereich – in einem harten Wettbewerb. Insbesondere mit dem nunmehr offenbar ernsthaften Einstieg IBMs in dieses Geschäft – mit dem System /6000 – wurden die Karten neu gemischt. Von größerem Interesse dürften nach der Vorstellung von Compaqs Systempro – s. MP 3/90 – bald die Multiprozessor-PCs werden; ganz zu schweigen von den Notebook-PCs, die als Ableger der Laptops immer mehr von sich reden machen. Da die Grenzen der einzelnen Computerklassen fließend sind, sollten Sie die Gliederung im Bericht nur als groben Rahmen sehen.

Workstations und Terminals

Nach ihrer bisher unbedeutenden Rolle im RISC-Geschäft (RISC – Reduced Instruction Set Computer; Rechner mit reduziertem Befehlssatz) drängt IBM nun offenbar ernsthaft auch in dieses Marktsegment. Gleich in mehreren Anwendungen präsentierte die Firma ihr neues **System /6000**, das als zweite Generation von RISC-Rechnern unter Unix neue Dimensionen eröffnen soll (Bild 1). Die Rechnerfamilie besteht insgesamt aus sechs Modellen, deren Leistungsspektrum von 27 bis 41 MIPS (Millionen Instruktionen pro Sekunde) reicht. Die hohe Systemleistung wird unter anderem durch den von IBM selbst entwickelten Mikroprozessor und die POWER (Performance Optimization With Enhanced RISC) genannte Architektur erreicht. Die Einstiegsversion im PC-Format ist das Modell 320 mit 27 MIPS bzw. 7,4 MFLOPS (Millionen Gleitkommaoperationen pro Sekunde), 8 bis 32 MByte Hauptspeicher und intern bis zu 640 MByte Plattenkapazität. Die Versionen 520, 530 und 540 sind Standmodelle, wobei letzteres mit 41 MIPS bzw. 13 MFLOPS das leistungsstärkste der Familie insgesamt ist. Ausgestattet wird es mit 256 MByte RAM in 4-MBit-Technologie. Mit einem integrierten Hochleistungsgrafikprozessor ist das Modell 730 vor allem für rechenintensive Anwendungen mit höchsten Grafikansprüchen beispielsweise in den Bereichen Simulation und Visualisierung gedacht. Zur Abrundung der Familie dient das Schrankmodell 930 mit intern 12 GByte Plattenkapazität und hoher Ausbaufähigkeit. So lassen sich über 500 asynchrone Ein- und Ausgabegeräte anschließen. Ergänzt wird das System durch das ebenfalls vorgestellte X-Terminal IBM Xstation 120, das im LAN über Ethernet oder Token-Ring mit AIX-, Unix- oder anderen Systemen, die das X-Window-Protokoll unterstützen, kommunizieren kann. Dabei übernimmt die Xstation bei X-Window-basierten Anwendungen die grafische Aufbereitung. Mit dem gleichfalls angekündigten neuen Betriebssystem AIX Version 3 soll ein Schritt zum Zusammenwirken der beiden IBM-Systemumgebungen SAA und AIX getan werden.
Weitere Ergänzungen der Produktpalette von IBM gab es mit der Vorstellung der PS/2-Modelle **65SX** (Prozessor 80386 SX, Standmodell, SCSI-Schnittstelle), Modell 80 (mit 80386/25 MHz und SCSI-Schnittstelle) sowie P70 386 (preisgünstigere Ausführung des portablen PS/2 mit 30 MByte Festplatte, 2 MByte RAM und 16 MHz Taktfrequenz).

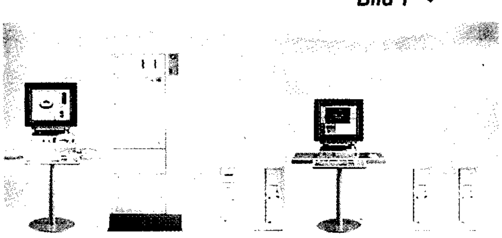
Evans & Sutherland, einer der führenden Anbieter von Hochleistungsgrafiksystemen und seit über 20 Jahren auf dem Markt, präsentierte seine neue 3D-Hochleistungs-Grafikworkstationfamilie **ESV** (Bild 2). Die Rechner basieren auf dem RISC-Prozessor R 3000 der Firma MIPS Computer Systems mit einer Leistung von 20 MIPS und sind in sechs Modellvarianten lieferbar. Der Anwender kann jedoch auch mit dem Einstiegsmodell beginnen und das System durch Hinzufügen von Platinen bis zum Spitzenmodell ausbauen. Das Einstiegsmodell ESV 5 bietet bereits die Darstellung von 360 000 Vektoren und 19 000 Polygonen je Sekunde; die ESV 50, das Topmodell, leistet 1 100 000 „Depth-cued"-Vektoren und 100 000 vierseitige, unabhängige Gouraud-schattierte Polygone pro Sekunde. Die Vektorleistungen werden durch in Hardware realisiertes Antialiasing, eine patentierte E & S-Entwicklung, erreicht. Die Gleitkommaleistung beläuft sich auf 4 MFLOPS (doppelte Genauigkeit) bzw. 8 MFLOPS (einfache Genauigkeit).
Alle Modelle laufen unter Unix System V mit BSD-Erweiterungen; berücksichtigt werden die Standards PHIGS (Programmers Hierarchical Interactive Graphics System), X-Window, MOTIF der OSF und PEX, die 3D-Grafikerweiterung zum X-Window-System, auf die bereits der Hardwareentwurf ausgerichtet wurde.

Intergraph stellte als Ablösung der bisherigen Modellreihe 3000 die neue **Serie 6000** vor. Die Firma nutzt für ihre Workstations den von Fairchild bzw. Intergraph entwickelten RISC-Prozessor Clipper C 300 mit Leistungen zwischen 10 und 14 MIPS. Mit dem neuen Grafikprozessor EDGE (extensible display graphics engine) können die volle 24-Bit-Grafik (16,7 Millionen Farben) und bis zu 400 000 Vektoren pro Sekunde erreicht werden. Auch ist eine Hardware-Antialiasing-Funktion vorhanden. In das Unix-Betriebssystem ist die Desktop-Benutzeroberfläche *Looking Glass* bereits standardmäßig eingebunden, womit das komplette Systemmanagement mittels Maus und Icons ermöglicht wird. Für den Benutzer entfällt damit die wenig anwenderfreundliche Unix-Kommandosprache.

Über die neue Tektronix-Workstation XD 88/10 hatten wir bereits in MP 4/90, S. 126, berichtet. Weitere Neuheiten waren daneben die Grafik-X-Station **XN7** (Bild 3) mit einer Workstation-ähnlichen Auflösung von 1280 × 1024 Punkten auf dem 19-Zoll-Schwarzweiß-Bildschirm und als kostengünstiges Farbmodell die Grafik-X-Station **XN10**. Die XN7 besitzt einen 15-MHz-68020-Prozessor zur Unterstützung anspruchsvoller technischer Anwendungen und acht serienmäßige Zeichensätze, die besonders beim DTP (Desktop Publishing) von Vorteil sind. Eine Ethernet-TCP/

Bild 2 ▶

Bild 1 ▼



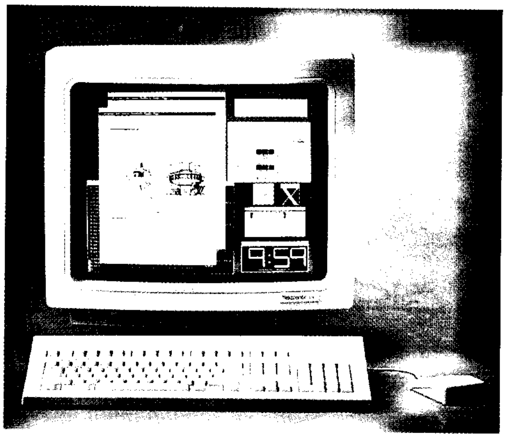


Bild 3

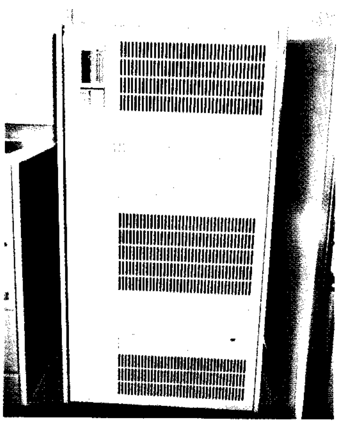
IP-LAN-Schnittstelle, ein RS-232C-Interface sowie Anschlußoptionen erlauben das Einbinden in heterogene Umgebungen mit Mainframes, Minis, Workstations und Terminals.

Die bisher als CTM (Computer Technik Müller) bekannte Konstanzer Firma stellte unter ihrem neuen Namen ITOS Computer GmbH die auf einem RISC-Prozessor der Firma MIPS basierende Workstation **ITOS 2000WS** (Bild 4) vor. Mit diesem Modell will man weniger auf den bestehenden Workstationmarkt abzielen als auf die künftigen Strukturen dieses Marktes. Dann nämlich wird nach Meinung von ITOS vor allem bei Großinstallationen die Workstation weitgehend zum PC-Ersatz. Und für eine solche massenhafte Verbreitung darf der Preis nicht zu hoch liegen. Mit weniger als 25000 DM als Komplettpreis der 2000WS (incl. Monitor mit 1280 × 1024 Punkten, Tastatur und 100-MByte-Festplatte) glaubt man sich bei ITOS gut im Rennen. Das Tischgerät läuft unter Unix V3 und leistet 14 MIPS.

Die Firma Digital Equipment GmbH München war mit ihrer bisher größten Offerte an der CeBIT beteiligt. Schwerpunkte der Ausstellung waren vor allem komplette Anwendungs- und Branchenlösungen, die sich auf Ausstellungsstände in acht Hallen verteilten. Im Rechenzentrum in Halle 1 wurden vor allem die neuen VAX- und RISC-Systeme gezeigt, die im zweiten Halbjahr 1989 auf den Markt kamen. Besonders erwähnt sei das Mehrplatz- und Serversystem **DECsystem 5810** (Bild 5) auf der Basis der RISC-Prozessoren MIPS R3000/R3010 unter dem Betriebssystem Ultrix (Unix-Betriebssystem von DEC). Die CPU-Leistung beträgt ca. 18 MIPS, der Hauptspeicher 32 MByte. Der externe Speicher kann bis auf 58 GByte ausgebaut werden. Die Erweiterung mit einem zweiten Prozessor und mit weiteren 32 MByte zum DECsystem 5820 ist möglich.

Die **VAX-6000-Familie** wurde um die Modelle 410 ... 460 erweitert, die mit 1 bis 6 Prozessorelementen ausgestattet sind. Jedes Prozessorelement der neuen Modelle hat eine Leistung von ca. 6,8 MIPS. Der Hauptspeicher kann auf 256 MByte ausgebaut werden. Die E/A-Leistung beträgt in Abhängigkeit von der Anzahl installierter BI-Bussysteme 20 bis 60 MByte/s. Für die Ausführung schneller E/A-Operationen wird ein Solid-State-Disk ESE20 mit einer Kapazität von 120 MByte angeboten. Die E/A-Leistung beträgt ca. 300 E/A-Gesuche/min und ist somit gegenüber Winchesterplatten zehnmal höher. Die elektronische Speichereinheit ESE20 ordnet sich voll in die Digital Storage Architecture (DSA) ein und ist somit clusterfähig.

Bild 5



Erstmals war das bereits angekündigte X-Window-fähige Terminal **VT1000** ausgestellt (Bild 6). Der Thin-Wire-Ethernet-Anschluß wird vom LAT- und TCP/IP-Protokoll unterstützt. Damit können die X-Window-Applikationen sowohl auf einem VMS- als auch Unix-Host laufen. Mit dem VT1000 wurde der neue superflache Elektrolumineszenz-Bildschirm **VREO1** gezeigt, der in Zusammenarbeit mit der Firma Planar Systeme entwickelt wurde. Der VREO1 hat eine Fläche von 394 × 343 mm^2 und ist nur 70 mm tief. Die Auflösung beträgt 1024 × 1280 Pixel und die Bildfrequenz 60 Hz. Er ist an jede Workstation anschließbar.

Gezeigt wurde u. a. auch die Anwendung von OSF/MOTIF unter DECwindows als grafisches Nutzerinterface, das die räumliche Simulation von Bedienelementen (Knöpfe, Schalter, Tasten etc.) gestattet.

Für die beschleunigte Grafikerweiterung wurde als neues Produkt die **VAXstation 3100 SPX** angeboten. Die VAXstation 3100 (Modelle 30 und 40) hat eine Leistung von etwa 2,4 MIPS. In Ergänzung gibt es die neueren Modelle 38 und 48 mit verbessertem CMOS-MicroVAX-Prozessor 78034M mit 3,8 MIPS. Der SPX-Grafik-Koprozessor (Scan Proc EXtension) ist für alle Modelle verfügbar. Er unterstützt 8-Bit-Ebenen (256 Farben). Die Grafikvektorleistung mit SPX erhöht sich um das Zehnfache.

Für spezielle Anwendungsfälle zeigten einige Hersteller neue fehlertolerante Unix-Rechner. Beispielsweise Tandem, allgemein als Marktführer bei ausfallgeschützten Parallelrechnern angesehen. Bei dem System **Integrity S2** (Bild 7) wird die Fehlertoleranz durch dreifach redundante RISC-CPUs mit jeweils doppeltem Hauptspeicher, doppelter Stromversorgung und doppelt angelegtem Ein-/Ausgabesystem gewährleistet (Triple Module Redundancy – TMR). Als Betriebssystem wird das Unix nach der System V Interface Definition (SVDI) von AT & T genutzt.

Als Vertragspartner von Tandem bietet Nixdorf dieses Konzept unter der Bezeichnung **Targon 3300** an. Der Rechner soll mittels TMR eine Verfügbarkeit von 99,997 Prozent erreichen. Das Betriebssystem basiert auf Unix System V Release 3.1 von AT & T, das um zusätzliche Funktionen zur Datensicherheit ergänzt wurde.

Ein weiteres ausfallgeschütztes System unter Unix System V (X/OS) kommt von Olivetti als Bestandteil seiner Open System Architecture (OSA) mit dem Minicomputer LSX 4030. Basis dieses Rechners ist jedoch ein Mikroprozessor Motorola 68030; der Hauptspeicher beträgt bis zu 32 MByte.

Personalcomputer

Im Mittelpunkt des Interesses standen sie zweifellos, die Spitzen-PCs mit dem neuen Intel-80486-Prozessor. PCs mit 80386 konnten da nur noch auffallen, wenn sie mit dem EISA-Bussystem ausgestattet wurden (von Zenith angekündigt); 286er PCs vielleicht, wenn man sie in ein exklusives Edelholzgehäuse steckte. Wie Panatek mit der „State of the art"-Edition des AT 80286 in einer limitierten Auflage von 300 Stück (Bild 8).

Mit der Vorstellung des Compaq Systempro begann sich eine neue Linie bei den Hochleistungs-PCs abzuzeichnen – hin zu Multiprozessorsystemen. Erwartungsgemäß stellte die taiwanesische Firma **Mitac** ihre **Serie 500** vor; standardmäßig mit zwei 80386/25 (später 80486) ausgestattet und auf sieben Prozessoren erweiterbar. Verbunden sind die Prozessoren in der Dual-Bus-Architektur mittels C-Bus, daneben gibt es noch den AT-Bus für Add-on-Karten und herkömmliche Peripherie. Bei voller CPU-Ausstattung können bis zu 160 Benutzer an das System angeschlossen werden. Verwendet wird das SCO Xenix System V als Betriebssystem. Bei einer Leistung des PCs von 25 MIPS glaubt Mitac, den VAX-Systemen von DEC und IBMs AS/400 den Kampf ansagen zu können.

Weiterer Anbieter von Multiprozessorsystemen war ITOS mit dem als Minirechner deklarierten Tower **ITOS 3020** (bis zu 4 Rechnerkarten mit je einem 68030-Prozessor und 16 MByte eigenem RAM; auf dem AT & T-Unix V 3.1 basierendes Multix).

Olivetti bietet – ähnlich einigen anderen Herstellern – besonders für wissenschaftliche Anwendungen und die Bildverarbeitung die Möglichkeit, den neuen 80486/25-PC optional zusätzlich mit Intels RISC-Prozessor 860 auszustatten.

Nach der Handvoll Aussteller von 80486-PCs zur Systems im Herbst '89 waren es derer nun bereits mehr als drei Dutzend. Die eindeutige Mehrheit davon stattete ihre Rechner übrigens mit dem EISA-Bussystem aus. Dies gilt auch für die ersten 80486-PCs mit 33 MHz Taktfrequenz. Beispielsweise Tandons neues Flaggschiff **486/33**. (Neben den sechs 32-Bit-EISA-Steckplätzen hat er auch zwei 16-Bit-AT-Slots, einen von 2 auf 64 MByte aufrüstbaren RAM sowie bis zu 670 MByte Fest-

Bild 4

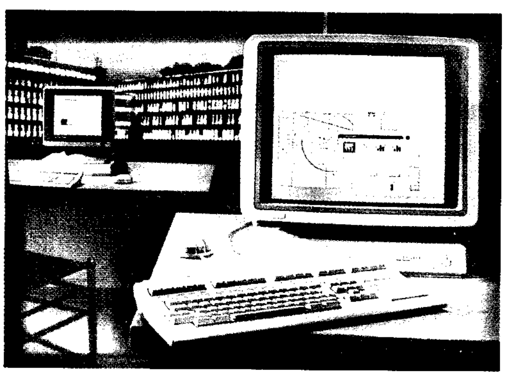


Bild 6

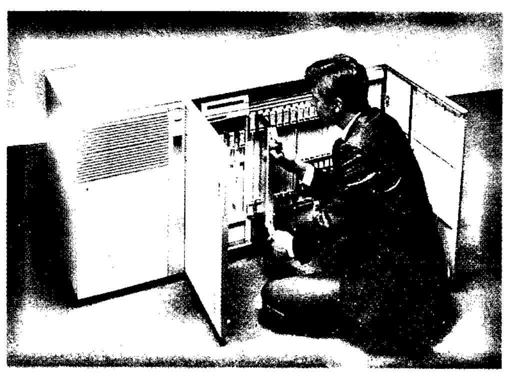


Bild 7

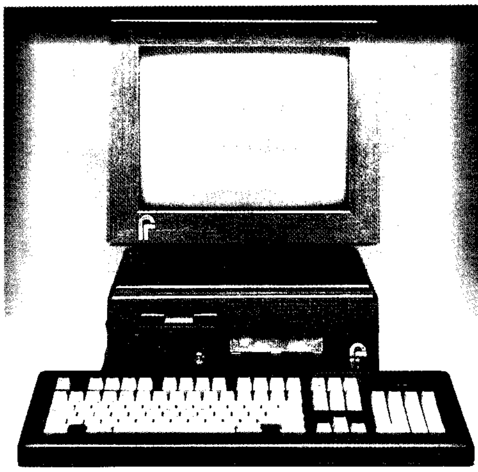


Bild 8 ▲

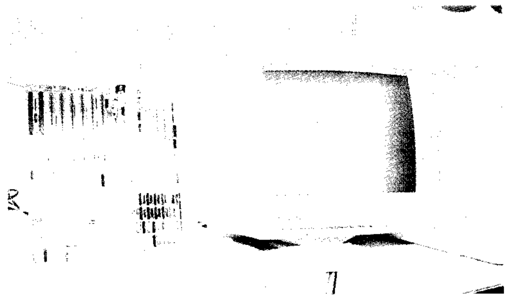
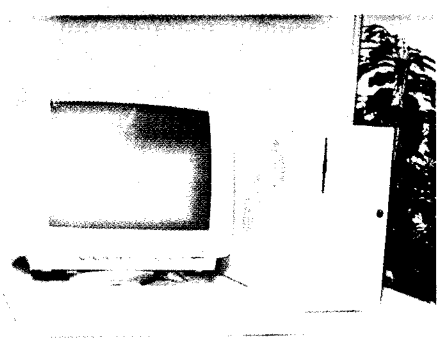


Bild 9 ▲

Bild 10 ▼



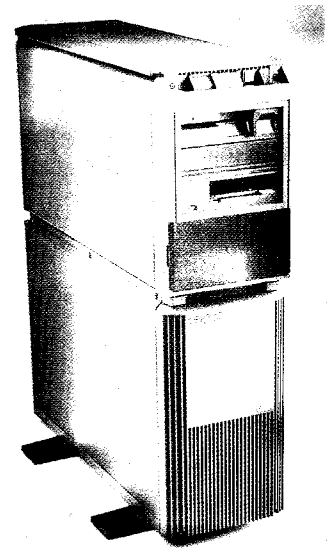


Bild 11

plattenkapazität.) Oder für den **EISA 486/33** von Dakota (Bild 9). Einen weiteren 33er zeigte der niederländische PC-Spezialist Laser Computer mit dem **Laser 486/1** (2 bis 16 MByte RAM, 8 bis 128 KByte Cache, 185-MByte-Festplatte). Die Mehrheit der Hersteller beschränkte sich allerdings auf die von Intel vorgegebenen 25 MHz, denn selbst da gab es bezüglich der Fehlersicherheit der Prozessoren noch einiges zu bemängeln.
Die Firma Schneider zeigte ihre bereits zur Systems präsentierte **486 Mega Station** (Bild 10). Sie steckt in dem vom Tower AT bekannten kompakten Gehäuse, in dem ein 3½-Zoll-Floppylaufwerk (1,44 MByte) und die 120-MByte-Festplatte untergebracht sind. Der RAM beträgt 2 MByte. An Schnittstellen sind vorhanden: 1 × seriell, 1 × parallel, 1 × Maus, 1 × externes Laufwerk. Die Auflösung des Monitors beträgt 1024 × 768 Bildpunkte (VGA).
Zur CeBIT-Halbzeit hatte Goupils neues 486er Flaggschiff, der **G60** Premiere (Bild 11). In dem markanten Tower haben zwei 5¼- oder 3½-Zoll-Diskettenlaufwerke, ein 150-MByte-Streamer und zwei Hochgeschwindigkeitsfestplatten (bis zu 2 × 1,2 GByte, SCSI-Controller) Platz. Dazu 8 Steckkarten im EISA/ISA-Format. Der RAM ist von 4 MByte bis auf 64 MByte ausbaubar. Goupil verzichtet auf einen externen Cache, da durch den Burst-Modus Speicherzugriffe ohne Wartezyklen möglich sein sollen.
AST war eines der ersten Unternehmen, die einen 486er vorstellten; nun gibt es fünf Modellreihen sowohl für den EISA- als auch für den ISA-Bus: AST Premium 486/33 (ISA), AST Premium 486/25 TE (Tower EISA), AST Premium 486/25 E(EISA), AST Premium 486/25T (Tower ISA) und AST Premium 486/25 (ISA). Allen Modellen gemeinsam ist die zum Patent angemeldete CUPID-32-Architektur (Completely Universal Processor, I/O-Design), bei der CPU, Speicher und Koprozessorunterstützung von den I/O-Funktionen, ASICs und dem BIOS auf der Systemplatine getrennt sind. Mit diesen Modellkonzepten wird die nachträgliche Aufrüstung zu mehr Leistung durch einfachen Austausch der Komponenten erleichtert.
Ebenfalls ein modulares Konzept verwendet Tulip bei dem neuen **tr 486e** (Bild 12); das heißt, der Prozessor sitzt nicht auf dem Motherboard, sondern selbst auf einer Steckkarte. Weitere Merkmale des tr 486e sind unter anderem der System Control Manager (SCM) als Sicherheitssystem (wurde in MP 12/89 bereits beschrieben), der EISA-Bus und das neue Tulip Advanced Memory Concept. Mit diesem werden die Schreiboperationen des Speichers optimiert, sprich um 30 Prozent gegenüber einem Second Level Cache reduziert, wodurch die Belegung des Busses durch den Systemprozessor drastisch verringert und der Bus für andere Benutzer frei wird. Tulip glaubt, daß der tr 486e mit diesem Konzept die beste und schnellste Lösung für den CAD-, DTP-, Fileserver- und Unix-Host-Markt darstellt.
Von den bekannten Anbietern von Personalcomputern mit Motorola-Prozessoren Apple, Atari und Commodore hatte lediglich Apple nennenswert Neues zu bieten. (Commodores Amiga 2500/30 wurde bereits zur LFM '90 gezeigt, s. MP 7/90, und der Amiga 3000 erst im April angekündigt; über die Neuheiten von Atari haben wir bereits berichtet.)
Neben IBMs System/6000-Vorstellung dürfte die Präsentation von Apples **Macintosh IIfx** (Bild 13) das zweite herausragende Ereignis der CeBIT gewesen sein. Immerhin soll der Neue mit der Steigerung der Taktfrequenz des 68030 auf 40 MHz bis zu viermal so schnell sein wie die Vorgänger IIcx und IIx (doppelt so schnell wie der IIci) und damit vielleicht der schnellste PC überhaupt. Neben der Erhöhung der Taktfrequenz wurden jedoch auch die anderen leistungsbeeinflussenden Systemmerkmale verbessert. Der Hauptspeicher ist mit 80-ns-DRAMs bestückt, der Cache mit 15-ns-SRAMs. Dazu kommen erstmals ein SCSI-Controller mit direkter Zugriffsmöglichkeit auf den Hauptspeicher (DMA) und zwei unabhängige I/O-Prozessoren (IOPs) für die „langsame" Peripherie. Gleichzeitig mit dem neuen Modell stellte Apple auch die neue Betriebssystemversion A/UX Version 2.0 vor, die auf Unix System V, Version 2.2 von AT & T basiert, sowie Videokarten einer neuen Generation. Die Aktiv-Videokarte 8 × 24GC besitzt beispielsweise einen Am 29000-RISC-Prozessor und soll ein bis zu 30mal schnelleres Arbeiten am Mac erlauben. Nach Apples Meinung ist der IIfx der erste PC, der Farbbildverarbeitung in Reproqualität erlaubt; mit den Möglichkeiten zur Echtzeitbildbearbeitung mit Tonsynchronisation, Farbdarstellung und Bildmanipulation soll er unter anderem eine ideale Entwicklungsplattform für Multimedia – übrigens ein neues Schlagwort auf der CeBIT – sein.

Tragbare PCs/Laptops

Bei den Entwicklungen der Hersteller von tragbaren Personalcomputern lassen sich gegenwärtig zwei Tendenzen erkennen: Erstens, die Laptops („Schoßcomputer" um die 6 kg Gewicht) an die Leistungen der Desktop-(Auftisch-)PCs heranzubringen und zweitens, die Computer soweit es nur geht zu verkleinern. Das führte zu einer neuen, mittlerweile als Notebook-PCs bezeichneten Kategorie, abgeleitet aus der Grundfläche eines Schreibblockes mit etwa A4-Format. Der weiteren Verkleinerung sind allerdings ergonomische, sprich natürliche Grenzen durch den Menschen gesetzt. Lassen sich doch schon die heutigen kleinsten Notebooks von Leuten mit Knubbelfingern kaum noch ernsthaft professionell bedienen. Bei der Leistungssteigerung konzentrieren sich die Bemühungen vor allem auf die Erhöhung der Speicherkapazitäten und die Qualität der Monitore, insbesondere auf die Darstellung von Farben.
In diesem Jahr sorgten drei Firmen mit Farblaptops für Aufsehen: Toshiba, Sharp und Hitachi. Bei Toshiba ging man den Weg, das bisherige Spitzenmodell, den T 5200, mit einem abnehmbaren 11-Zoll-Farbdisplay auszustatten. Das LCD mit passiver Darstellungsmatrix am gezeigten Vorserienmodell des **T 5200 C** (Bild 14) stellt bis zu 16 Farben dar und hat eine Auflösung von 640 × 480 Bildpunkten (VGA). Die geringe Bildpunktgröße (0,105 × 0,335 mm²) und der geringe Punktzwischenraum (0,015 × 0,025 mm²) sorgen dabei für

Bild 13 ▶

Bild 12 ▼

die gute Bildqualität. Der T 5200 C arbeitet mit einem Prozessor 80386, 20 MHz; er hat eine 100-MByte-Festplatte und 2 Erweiterungssteckplätze.

Die Firma Hitachi, die Anfang 1989 erstmals einen Laptop mit Farbdisplay präsentierte – allerdings mit einer bescheidenen Diagonale von 6,3 Zoll (s. MP 4/89, S. 124) – zeigte nun erstmals ein Modell mit 10-Zoll-Monitor. Der **HL 500C** (Bild 15) arbeitet mit einem 80386SX-Prozessor, hat 1 MByte RAM, eine 3,5-Zoll-Floppy und eine 40-MByte-Festplatte. Bei dem Farb-LCD mit einer Auflösung von 640 × 480 × 3 Punkten und acht darstellbaren Farben verwendet Hitachi die sogenannte Dünnfilmtransistortechnik (TFT – Thin Film Transistor), bei der die Bildpunkte durch im Bildschirm integrierte Transistoren gebildet werden (das Prinzip hatten wir bereits in MP 1/90, 2. Umschlagseite, ausführlich erläutert). Damit sind besonders schnelle Bildwechsel ohne Schlierenbildung möglich.

Auf dem gleichen Monitorprinzip beruht das 10-Zoll-LCD des neuen **Farblaptops von Sharp** (Bild 16). Die Firma hatte bereits zur CeBIT '89 den ersten 386-Portable mit einem 14-Zoll-Farbdisplay präsentiert (s. MP 6/89, 4. US); nun ließ sie den Prototyp ihres neuen Modells für die CeBIT eigens aus Japan einfliegen. Auch Sharps Glanzstück beeindruckte durch die hohe Auflösung (640 × 480 × 3), die klare und kontrastreiche Darstellung und den mehr als 60° großen seitlichen Blickwinkel.

Für Laptops im Moment noch Zukunftsmusik, aber vielleicht bald verfügbar sind die von IBM vorgestellten 10-Zoll-LCDs. Während einer der Prototypen mit jedem der 640 × 480 Pixel 512 Farben darstellen kann, demonstrierte der andere Bildschirm die Möglichkeit von jeweils 262 144 verschiedenen Farben. Dazu war er mit einer auf ein Aquarium gerichteten Videokamera gekoppelt, deren „beruhigende" Bilder quasi in Echtzeit übertragen wurden.

Einen Blick in die Zukunft erlaubte auch Toshiba mit einem 14-Zoll-TFT-LCD. Das Display, das wie IBMs Entwicklungen nicht nur für Laptops, sondern auch für die Unterhaltungselektronik interessant ist – Schlagwort Multimedia –, kommt von der Firma Display Technology, Inc., einem Gemeinschaftsunternehmen von Toshiba und IBM, und ist das erste Ergebnis der Kooperation. Als Besonderheit besteht dieses Display aus je vier einzelnen Zellen bzw. Punkten für die Farben. Das heißt, zu rot, grün und blau kommt noch eine weiße Zelle, die dem Bild einen höheren Kontrast verleiht.

Neben den Bemühungen der Hersteller, Farbe auf die „Transportablen" zu bringen, gibt es auch andere interessante Entwicklungen. Die Firma Dolch beispielsweise stellte erstmals in Europa den ersten transportablen PC mit 80486-Prozessor vor (wir berichteten über den P. A. C. 486/25 bereits in MP 4/90, S. 126); Goupil präsentierte als erster – mit dem Golf SX – einen Laptop mit Intels brandneuem 20-MHz-Prozessor 80386SX/20; REIN Elektronik bot mit dem M5 einen Laptop mit modularem Aufbau an, das heißt mit der Möglichkeit, verschiedene Komponenten nach Bedarf zu nutzen (das Konzept ist allerdings bereits von Epsons PX-16 bekannt; s. MP 9/89, 4. US).

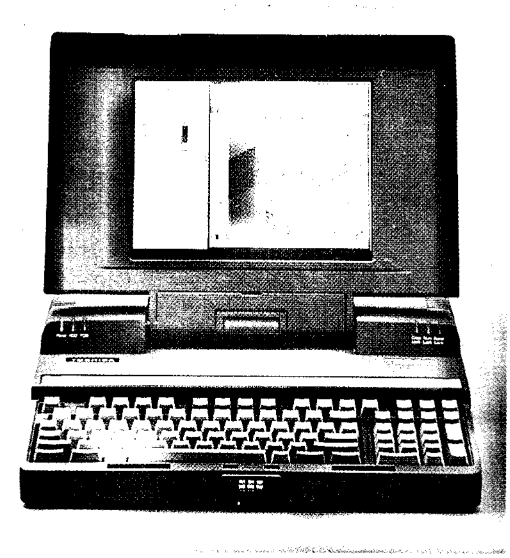


Bild 14

Bild 15

Notebook-PCs

Auf dem Gebiet der „Mini-Laptops" tummelt sich mittlerweile eine Vielzahl von Herstellern mit immer neuen Produkten. Toshiba, mit dem T 1000 einer der ersten, bot zahlreiche Weiterentwicklungen an (T 1000 SE, T 1000 XE, T 1200 XE); natürlich waren Sharp mit dem neuen PC-6220 (s. MP 5/90, S. 158), Compaq mit dem LTE (s. MP 1/90, S. 30) und Atari mit dem Portfolio (s. MP 12/89, S. 384) vertreten. Beim Vergleich dieser MS-DOS-PCs läßt sich bereits erkennen, daß in die Gruppe der „Notebooks" doch recht verschiedene Konzepte fallen. Die Größe liegt teilweise deutlich unter A4-Grundfläche, teilweise aber auch darüber. Dementsprechend sind einige Geräte mit herkömmlichen Festplatten- und Diskettenlaufwerken ausgestattet, andere wegen der geringen Größe nur mit Speicherkarten. Ein typischer Vertreter ist von Philips der neue **PCL 101** (Bild 17). Seine Maße betragen 28 × 22 × 2,8 cm³ (B × T × H) und das Gewicht 2,3 kg. Er arbeitet mit einem 80C86-Prozessor, ist also XT-kompatibel. Der PCL 101 besitzt ein 3,5-Zoll-Floppylaufwerk, das Betriebssystem ist im ROM gespeichert. Das LC-Display hat CGA-Auflösung (640 × 200 Punkte) und kann acht Graustufen darstellen. Ein Batteriesatz soll für drei Stunden Betrieb sorgen.

Bis zu 60 Stunden Betriebsdauer mit einer Batterieladung gibt die PSION GmbH für ihre **Psion-MC-Serie** an (MC 200, MC 400, MC 600). Erreicht wird das vor allem durch die Verwendung von Intels Flash-Eproms als RAM.

Als vielleicht kleinster voll funktionsfähiger MS-DOS-Rechner sei schließlich noch der **Poqet PC** der Poqet Computer Ltd. erwähnt – nur 22 × 11 × 2,3 cm³ „groß" und 450 Gramm schwer. Neue Energiespartechnologien und Speicherkarten erlauben mit zwei Mignonbatterien einen Betrieb von bis zu 100 Stunden. Diesen interessanten PC werden wir demnächst etwas ausführlicher vorstellen.

Peripherie

Insbesondere bei Disketten- und Festplattenlaufwerken dürften die Entwicklungen – in Wechselwirkung – nicht unwesentlich von der weiteren Miniaturisierung der PCs geprägt sein. So setzt Toshiba in den neuen T 1200XE-Laptop bereits ein 2,5-Zoll-Festplattenlaufwerk mit 20 MByte des amerikanischen Herstellers Conner Peripherals ein. Die gleiche Firma bietet für das herkömmliche 3,5-Zoll-Format bei einem Zoll Bauhöhe 120 MByte Speicherkapazität bei 19 ms mittlerer Zugriffszeit (Conner CP-30100). Beim 5,25-Zoll-Format werden vom Seagate Wren Runner 767 MByte erreicht, vom Solid Computer Disk 1200 1,2 GByte und beim D 5892 von NEC 1,6 GByte. 8-Zoll-Festplatten bringen es auf Kapazitäten von 2,65 GByte (Fujitsu M 2671P).

Eine ähnliche Tendenz zeichnet sich bei Diskettenlaufwerken ab: Erhöhung der Speicherkapazitäten und Datentransferraten, Verringerung der Zugriffszeiten und gleichzeitig Versuche, neue, kleinere Formate auf dem Markt durchzusetzen. Allgemein wird eingeschätzt, daß die bisher beherrschende Stellung der 5,25-Zoll-Disketten gegenüber dem 3,5-Zoll-Format demnächst verloren geht. Als neuer Kapazitätsstandard zeichnen sich bei 3,5-Zoll-Disketten bereits 4 MByte ab. Sollte sich das von Insite Peripherals entwickelte Floptical-Prinzip verbreiten, kann bald mit 20-MByte-Floppies in den PCs gerechnet werden.

Die Druckerhersteller nutzten die CeBIT einmal zur Vorstellung von Weiterentwicklungen bewährter Produkte. Beispielsweise NEC mit den Nachfolgern der bekannten Matrixdrucker Pinwriter P6 und P7, jetzt **P60** und **P70** genannt; Seikosha mit der schmalen Version des SL-230, dem neuen Modell **SL-210**; Hewlett-Packard mit dem Nachfolger des HP IIP, jetzt **HP III**; C. ITOH mit der Breitversion des Matrixdruckers C-610, **C-645**. Der C-645 ist in zweierlei Hinsicht bemerkenswert. Zum einen ist es der erste 28-Nadeldrucker der Welt. Die Nadeln sind dabei im Druckkopf in drei Reihen zu je 9 Nadeln plus

◀ *Bild 16*

Bild 17 ▲

einer zehnten Nadel in der mittleren Reihe angeordnet (Bild 18). Zum anderen gehört er zu den Flachbettdruckern, einer Bauweise, die wegen des waagerechten Papierdurchzuges Vorteile besonders beim Bedrucken von Briefumschlägen, Etiketten, Aufklebern, dickem Papier oder mehreren Durchschlägen bietet. Drucker der gleichen Bauweise wurden von Epson gezeigt (24-Nadler, ohne Typenbezeichnung) und mit dem **SL-532** von Seikosha (vorgestellt auf der 4. Umschlagseite dieses Heftes). Epson dürfte der Hersteller mit der vielleicht größten Anzahl von Druckerneuheiten zur CeBIT gewesen sein. So gab es neben dem neuen Flachbettdrucker unter anderem den als „Geschwindigkeits-Weltmeister" präsentierten 18-Nadler **DFX-8000** zu sehen. Im High Speed Draft Modus soll er 1066 Zeichen/Sekunde schnell sein, bei NLQ und 10 cpi 160 Zeichen/Sekunde. Mit dieser Leistung und der angebotenen Ausstattung eignet sich der DFX-8000 neben PC-Arbeitsplätzen vor allem für den Bereich der mittleren Datentechnik.
Als „ersten echten seitenorientierten Drucker auf Tintenstrahlbasis" offerierte Epson schließlich einen **64-Düsen-Tintenstrahldrucker** (ohne Typenbezeichnung, Bild 19). Für die komplexe, also seitenorientierte Darstellung ist er von standardmäßigen 0,5 MByte intern bis auf 2,5 MByte RAM aufrüstbar. Als Vorteil gegenüber den meisten Laserdruckern verarbeitet er auch A3-Papier und Leporello (Endlospapier). Er bietet die auch bei Laserdruckern übliche Auflösung von 300 dpi (Punkte pro Zoll), ist allerdings nur etwa 1 Seite/Minute schnell bei Letter Quality.
Als weitere Neuheiten seien abschließend genannt: der erste Thermotransfer-Farbdrucker für OS/2-Anwendungen, der Writer PM 10 von QMS, der „Triathlon", ein Nadeldrucker der Schweizer Firma Wenger, bei dem je nach Bedarf ein 18- oder 24-Nadeldruckkopf eingesetzt werden kann („von jeder Sekretärin in wenigen Minuten erlernbar") sowie von Pentax der erste Laserdrucker, der auch Leporellopapier verarbeitet.

Im Grafikbereich setzte auf der CeBIT die Mehrzahl der Firmen auf VGA (Video Graphics Array) mit der Standardauflösung von 640 × 480 Bildpunkten bei 16 Farben. Zum guten Ton gehörten aber auch schon die Auflösungen 800 × 600 (Super-VGA) und 1024 × 768 (z. B. VESA, vgl. MP 3/90, S. 95) bei 4 bis 256 Farben. Nur noch selten war EGA (Enhanced Graphics Adapter) mit 640 × 400 Bildpunkten bei 8 Farben zu finden.
Mit steigender Auflösung und Anzahl der Funktionen ist zwar mehr „Intelligenz" integriert, jedoch läßt sich immer weniger ein neuer Standard erkennen. Da reichen die Auflösungen jenseits von VGA von 1024 × 1024 Pixeln bis zu Auflösungen von 1600 × 1280 oder 1664 × 1200 Pixeln; dabei können bis zu 256 Graustufen oder 256 Farbschattierungen pro Farbe (= 16,7 Mio Farben) erzielt werden. Auch bei den Schnittstellen schien sich (neben Apple) für die PCs noch kein Standard durchgesetzt zu haben. Der bereits 3 Jahre alte 8514/A-Adapter mit dem *Application Interface* (AI) von IBM hat vom *Texas Instruments Graphics Adapter* (TIGA) Konkurrenz bekommen. Ein Beispiel dafür ist die **Hercules Graphics Station Card** (Bild 20) der Kalifornischen Firma Hercules Computer Technology. (Hercules ist es bislang als einzigem Hersteller von Grafikkarten gelungen, neben IBM einen Standard durchzusetzen.) Die Karte verwendet den VGA-Controller 82706 von Intel, den mit 60 MHz getakteten Grafikcontroller 34010 von Texas Instruments, bis zu 2 MByte RAM und 1 MByte Video-RAM. Erzeugt werden damit 1024 × 768 Bildpunkte mit 256 Farben (bei 8 Bit pro Pixel) oder 16,7 Millionen Farben (bei 24 Bit pro Pixel). Ihr Preis beträgt rund 2500 Mark.

Auf der Basis von Transputern der Firma Inmos bot der Aachener Hersteller von Parallelrechnern *paracom* den Grafikmodul **GDS II** an. Für alle wichtigen Rechnertypen anpaßbar, erreicht er eine Auflösung von bis zu 1280 × 1024 Pixeln. Bei einer Speicherung von 24 Bit pro Pixel können ebenfalls 16,7 Millionen Farben gleichzeitig dargestellt werden (Bild 21).

Bei den Monitoren ist ein Trend zu größeren Durchmessern (bis zu 21 Zoll) und zu verbesserter Ergonomie zu verzeichnen. Viele Endprodukthersteller, insbesondere in Europa, setzen zudem nur noch strahlungsarme Monitore ein. Beispielsweise besitzt der 14-Zoll-Monitor **TDV 1472-C** von Tandberg Data (Norwegen) neben einer Bildfrequenz von 70 Hz und dem bekannten ergonomischen Tandberg-Farbkonzept (vgl. MP 6/89) eine interne Metallverkleidung, die sowohl die Strahlung auf ein niedriges Niveau senkt als auch den Einsatz von toxische Gase enthaltenden Flammenhemmern im Kunststoffgehäuse unnötig macht.

Der 15-Zoll-Bildschirm **IRIS 15s** der Erkrather Firma *egs* erreicht im Super-VGA-Modus (800 × 600) eine Bildfrequenz von 73 Hz und im EGA-Modus sogar von 121 Hz! Damit wird eine völlig flimmerfreie Anzeige möglich. Das Gehäuse wurde unter Verzicht auf Plastmaterialien aus Metall gefertigt. Zur Grundausstattung zählen ebenfalls ein höhenverstellbarer Fuß und die stufenlose Verstellung des Neigungswinkels (Bild 22). Sein Preis wird mit rund 5000 Mark angegeben.

Völlig strahlungsfrei sind LC- und Elektrolumineszenzdisplays. Ein schon seit Jahren erfolgreicher Entwickler der aktiv leuchtenden EL-Diplays ist die Firma FINLUX Terminal – eine Tochter der Lohja Corporation

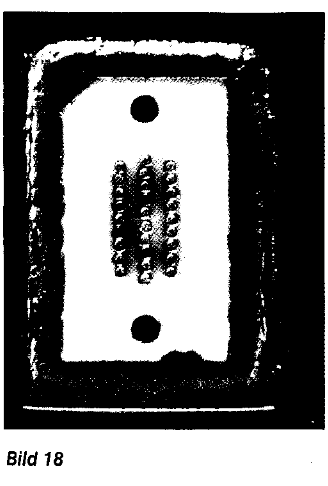
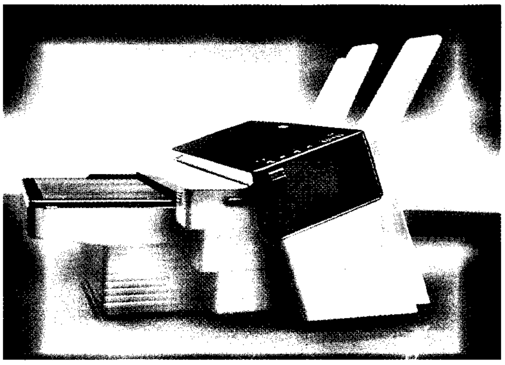


Bild 18

Bild 19



Finnland. Sie stellte zur CeBIT als „Weltneuheit" das VGA-kompatible 11-Zoll-Display **ELM640×480** vor (Bild 23). Es hat eine Bildfrequenz von 60 Hz im Grafikmodus (640 × 480) und von 70 Hz im Textmodus (640 × 400). Das Display kostet rund 3800 Mark.
Die verbesserten Grafikmöglichkeiten der PCs lassen natürlich die Frage aufkommen, ob man nicht beispielsweise auch Fernsehen oder Bildtelefon über Computer realisieren kann. Die Antwort heißt Multimedia oder *Digital Video Interactive* (DVI). Als Beweis bot Toshiba zum erstenmal die Möglichkeit, mit einem Laptop fernzusehen, das heißt Videofilme in Gesamtlänge mit Trickbildern, 3D-Grafiken, Text und hochwertigem Stereoton zu vereinen. Hierzu wurden neue Speicherverfahren entwikkelt, denn bisher wurden etwa 22,5 MByte für die Darstellung von nur einer Sekunde eines Farbfernsehbildes benötigt. Mit Hilfe spezieller Algorithmen und dem leistungsfähigen Mikroprozessor 82750 hat Intel ein Verfahren zur Verdichtung der riesigen Informationsmengen entwickelt. Dieses Verfahren ermöglicht erstmals eine Echtzeit-Dekompression, das heißt, die komprimierten Daten werden in Echtzeit gelesen und in das gewünschte Format umgewandelt. Es ist jetzt möglich, daß mehr als eine Stunde normaler Film mit 512 × 480 Bildpunkten, 30 Bildern pro Sekunde und Stereoton auf einer einzigen 5-Zoll-CD-ROM gespeichert werden kann.
Ein anderes Beispiel bot die Nürnberger Firma DIGIHURST mit der Integration von Text, Bild, Ton und Diagrammen im **MicroEye PictureBook Board** (Bild 24). Aber auch IBM gehört bereits zu den Multimedia-Anbietern – hier **IBM AVC** genannt (Audio Video Connection). Auf der Basis des Personal System/2 werden Bilder, Grafiken, Audio und Videoelemente in einer Präsentation vereint.

Netzwerke

Für den Messebesucher schwieriger zu entdecken als ein PC mit Farbmonitor ist ein Netz, an das der PC angeschlossen ist. Zumal die Vielfalt der Vernetzungen auf der CeBIT '90 ähnlich schwer zu überblicken war wie die der PCs. Der CeBIT-Schwerpunkt Netzwerke mit 154 Ausstellern in den Hallen 14 und 23 enthielt unter anderem die großen system- und herstellerübergreifenden Gemeinschaftsveranstaltungen NetWorld Europe, MultiNET und EurOSInet.
Obwohl die NetWorld mit 65 beteiligten Firmen in Halle 23 das größte CeBIT-Netz vorstellte, war doch die MultiNET wesentlich überschaubarer: In der Mitte der Halle 14 stehend, konnte man alle 25 Netzteilnehmer leicht überblicken. Auch die Vielfalt der vorgestellten Hard- und Softwarekomponenten ließ interessante

Bild 20

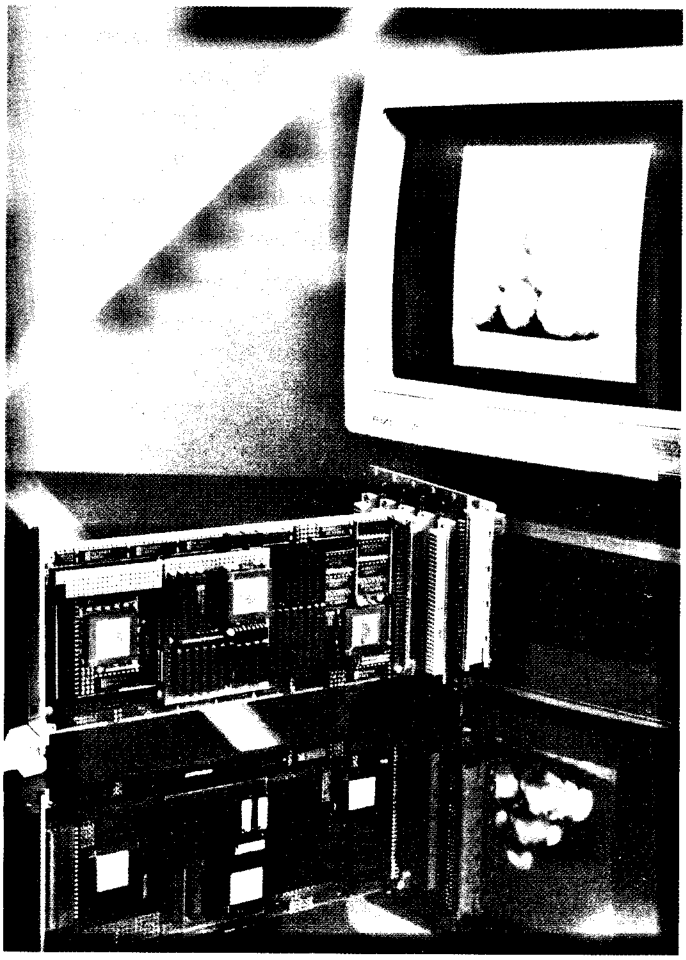


Bild 21

Trends erkennen. So wurde auf der MultiNET '90 eine herstellerübergreifende FDDI-Vernetzung vorgeführt, die in dieser Art zum ersten Mal in Europa zu sehen war. Angeschlossen an das FDDI waren Teilnetze wie Ethernet und Token Ring.
Die MultiNET, die seit 1987 stattfindet, wollte anfangs lediglich die Möglichkeit der Kommunikation von Produkten verschiedener Hersteller über standardisierte Basisdienste demonstrieren. In diesem Jahr wurden dagegen verstärkt Anwendungen präsentiert, die auf verschiedenen Systemen lauffähig sind und Aufgaben untereinander aufteilen können. Wohl am wichtigsten erscheint die Demonstration der Zusammenarbeit der traditionellen Netze (Ethernet mit 10 MBit/s und Token Ring mit 4 bzw. 16 MBit/s) mit dem neuen FDDI (Fiber Distributed Data Interface mit 100 MBit/s), das sich noch in der Normierungsphase befindet. Es ermöglicht den Anschluß von bis zu 1000 Stationen. Die maximale Ringlänge der optischen Glasfasern beträgt 200 km bei einem Abstand von maximal 2 km zwischen zwei Netzknoten. Es eignet sich durch seine sehr hohe Geschwindigkeit und auch wegen des noch sehr hohen Preises in erster Linie für eine leistungsfähige gebäudeübergreifende Vernetzung. Für die Zukunft wollen es Experten jedoch als das „Ethernet der 90er Jahre" – also auch für die Inhouse-Vernetzung – sehen. Der Begriff „Ethernet" wird aber nur deshalb benutzt, weil das Ethernet mit seiner Linienstruktur die größte Verbreitung hat; das FDDI ist nämlich ein kollisionsfreies Netz mit einer Ringstruktur, das nach dem Token-Verfahren arbeitet.
Für die Verbindung unterschiedlicher Netze werden natürlich leistungsfähige Verbindungsbaugruppen – je nach Einsatzfall Bridges oder Router – benötigt. Einen solchen Router für die Kopplung von Ethernet und FDDI zeigte die Karlsruher Firma Schneider & Koch. Er basiert auf einem PC/AT mit den Komponenten **SK-NET FDDI-Board**, SK-NET G16 Ethernet-Controller und Router-Software. Bis Ende des Jahres soll hierfür auch ein Token-Ring-Controller verfügbar sein. Der FDDI-Controller (Bild 25) ist mit dem Signalprozessor AMD 29000 und dem Super-Net-Chipsatz von Advanced Micro Devices bestückt. Er enthält 1 MByte RAM und leistet bei einer Taktfrequenz von 20 MHz 17 MIPS (Millionen Instruktionen pro Sekunde). Die Lichtwellenleiter-Interfaces ermöglichen eine gegenläufige Doppel-Ring-Struktur (Class A), die mit ihrer Redundanz eine hohe Zuverlässigkeit garantiert.

Mit der **ISOLAN FDDI/802.3 Bridge** zeigte der englische Netzwerkhersteller BICC Data Networks auf der CeBIT '90 sein erstes Produkt der ISOLAN-Familie, das voll dem FDDI-Standard für Hochgeschwindigkeits-Glasfasernetze entspricht (Bild 26). Noch in diesem Jahr soll bei BICC Data Networks wie auch bei SK ein entsprechendes Token-Ring-Produkt folgen. Die Verbindung von BICC Data Networks in Halle 23 und Siemens in Halle 1 über einen FDDI-Ring war nicht zufällig. Siemens beabsichtigt, mit der Verarbeitung von LAN-Komponenten von BICC von Anfang an beim FDDI-Boom dabei zu sein.

Da die Möglichkeiten der Vernetzung von Rechnern nicht so einfach erfaßbar sind, ließ sich die Firma Novell etwas ganz besonderes einfallen. Sie vernetzte 250 Workstations von Acer, Apple, AST, Compaq, HP, IBM, Olivetti, Philips und Tulip über ein Ethernet und stellte sie an einer großen Videowand auf. Als Server diente ein einziger Compaq Systempro 386 mit EISA-Bus, 33 MHz, 16 MByte RAM und dem Netzwerkbetriebssystem **NetWare 386** von Novell.
Eindrucksvoll wurde demonstriert, daß die einzelnen Workstations sich die Aufgaben – in diesem Fall die Bilder – untereinander aufteilen können. Drei Betriebssysteme (DOS, OS/2, Mac) und 10 Applikationen (z. B. Turbo-C, Norton Utilities, PageMaker, Foxbase und MS-Word) „spielten" mit. Damit konnte auch der Trend zu den Client-Server-Architekturen aufgezeigt werden – also nicht nur Ressourcenteilung für viele Anwender, sondern auch Arbeitsteilung zwischen Workstation und Server.

Auf dem Softwaresektor konnte – insbesondere auf der MultiNET – die Koexistenz von TCP/IP und ISO-Protokollen beobachtet werden. Aber hier ist es wie bei den Betriebssystemen (proklamierte Ziele sind OS/2 oder Unix, aber fast alle arbeiten mit DOS): Das *Transmission Control Protocol/Internet Protocol* prägt immer noch sehr stark die offene Kommunikation, obwohl es das erklärte Ziel aller ist, das 7-Schichten-Modell der Internationalen Standardisierungs-Organisation zu verwenden.
Netzwerkbetriebssysteme wurden unter anderem von Microsoft mit den LAN-Managern 2.0 und /X, von IBM mit dem LAN-Server, von Banyan mit Vines 4.0 und von Novell mit der oben erwähnten NetWare 386 angeboten.

Software

Betriebssysteme und Benutzeroberflächen

Ganz deutlich zeigte sich, daß die erstmals auf dem Apple-Macintosh eingeführte, nutzerfreundliche Bedienoberfläche immer mehr zu allgemeinem Standard wird und viele Softwareprodukte über Ikonen oder Menüs gesteuert werden können. Auch Unix, in der Vergangenheit eines der am schwierigsten zu bedienenden Betriebssysteme, kommt um diese Entwicklung nicht herum. Dies kann eine Voraussetzung sein, damit Unix vielleicht doch noch zu dem am weitesten verbreiteten Betriebssystem für Mikrocomputer wird, was gegenwärtig „nur" für Mehrplatzsysteme zutrifft. Hierbei allerdings soll der Anteil bei weit über 50 Prozent liegen. Handicap für den Einsatz von grafischen Bedienoberflächen auf Mikrocomputern unter Unix waren bisher die Hardwarevoraussetzungen: mindestens ein 80386er Prozessor (besser ein 80486er) und ein Hauptspeicher mit einer Größe ab 8 MByte. Mit der faktischen Aufhebung der COCOM-Bestimmungen für die DDR seit Juli dürfte aber auch dies keine große Beschränkung mehr sein. Für das Gebiet der DDR ist aber aus einer anderen Sicht mit einem Anstieg der installierten Unix-Systeme zu rechnen.
So hat die Bundesregierung bereits im Januar 1990 für alle (bundes-) deutschen Ministerien, für die Bundesbank und die Bundesbahn die Übernahme des X/Open-Standards empfohlen.

Bild 22

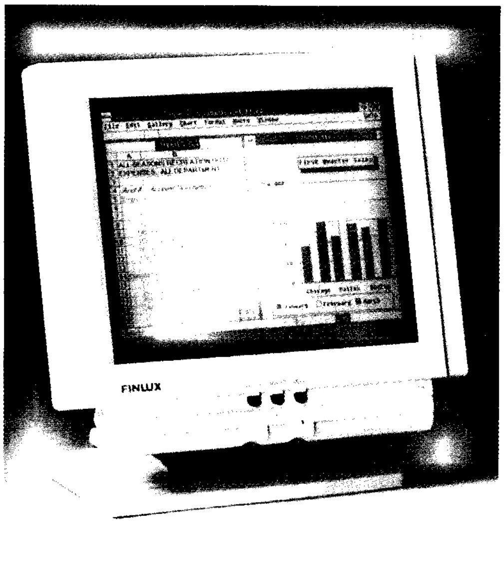


Bild 23

Bild 24 ▲

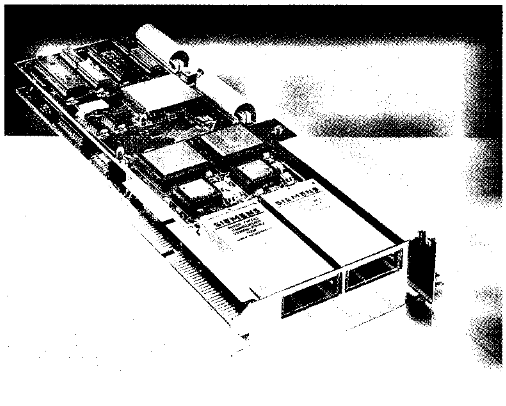


Bild 25 ▲

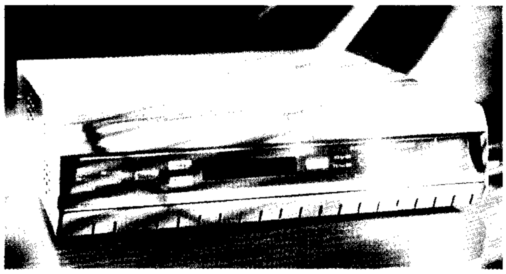


Bild 26 ▶

Im Jahre 1984 wurde **X/Open** als ein unabhängiges, gemeinnütziges Konsortium gegründet, dessen Mitglieder sich verpflichten, hinsichtlich ihrer Hard- und Software den Definitionen der einheitlichen Anwendungsumgebung (**C**ommon **A**pplication **E**nvironment) Rechnung zu tragen. Die Weiterentwicklung dieser Umgebung erfolgt jedoch nicht nur durch die X/Open-Mitglieder, sondern in enger, weltweiter Zusammenarbeit mit Anwendern, unabhängigen Softwareanbietern und Standardisierungsorganisationen. Der aus dieser Entwicklung resultierenden Rolle der sogenannten offenen Systeme wollen wir in Zukunft dadurch gerecht werden, daß wir einzelne, ausgewählte Softwarepakete in unseren Rubriken *vorgestellt* und *kennengelernt* ausführlicher beschreiben werden als das im Rahmen einer Messenachlese möglich wäre. Vorgesehen ist beispielsweise die grafische Nutzeroberfläche **Open Desktop** der Santa Cruz Operation (SCO).

Das System- und Softwarehaus GEI ermöglicht mit der neu angebotenen **Uniface-Version 5**, einer Sprache der 4. Generation (4GL), den Zugriff auf alle Daten eines Unternehmens aus einer Anwendung heraus, der auch unabhängig vom Betriebssystem, von der Datenbank, von dem Netzwerk oder der Benutzeroberfläche möglich sein soll. Für Uniface wurden im März 1990 die X/OPEN Application Labels vergeben, was bedeutet, daß nicht nur Uniface-Anwendungen, sondern auch Uniface selbst der einheitlichen Entwicklungsumgebung (CAE) gerecht werden.
Eine interessante Lösung scheint auch **Eurix**, eine von Generics Karlsruhe angebotene, deutschsprachige Implementation der Version 3.2 von Unix V/386 zu sein, die die Akzeptanz von Unix auch unter *Nicht*-Programmierern erhöhen könnte.

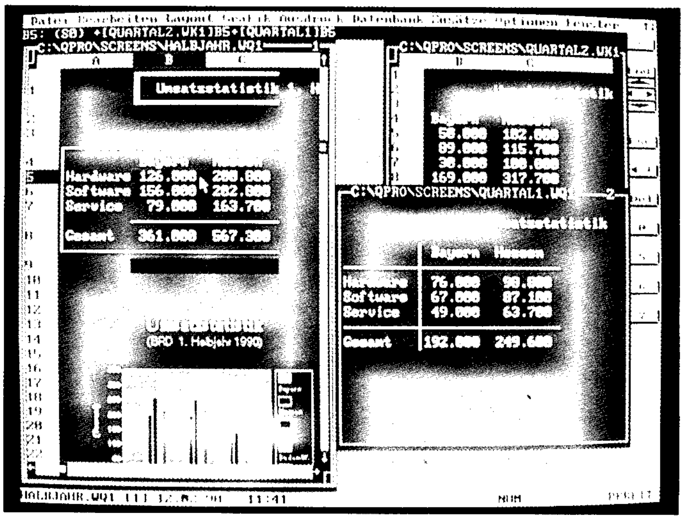
Digital Research, Altmeister der Betriebssysteme für Mikrocomputer, hat sich offensichtlich vom Tiefschlag durch den IBM PC mit MS-DOS erholt und bietet neben dem zu MS-DOS 3.30 kompatiblen Betriebssystem **DR DOS 3.41** für Einzelplatzsysteme mit **Concurrent DOS** auch ein Multiuser-/Multitaskingbetriebssystem an. Wir stellen Ihnen im Heft 9/90 neben anderen auch dieses System etwas ausführlicher vor. Von sich reden macht Digital Research aber auch durch die grafische Benutzeroberfläche **GEM/3 Desktop** und das Grafikprogramm **Artline 2**, das auch den Ansprüchen von Designern und Grafikern sowie den Anforderungen des Verlagswesens genügen soll. Beliebige Definition der Farben und der Farbverläufe und auch die Ablage in sogenannten Farbpaletten im RGB- (rot-gelb-blau), CMY- (cyan-magenta-yellow) oder HLS-Farbmodell (hue-lightness-saturation/Schattierung-Helligkeit-Sättigung) sind nur einige der Möglichkeiten, die Artline 2 bietet. Die Darstellung der 16 Millionen Farben ist lediglich von der Qualität der Ausgabegeräte abhängig.
36 Outline-Schriften gehören zum Lieferumfang und entsprechen damit einem Standard-Postscriptdrucker, die Vierfarbseparation ist möglich. GEM/3 Desktop und die dafür speziell entwickelten Ausbildungspakete sollen in Zukunft, so eine Vereinbarung zwischen dem Bayerischen Kultusministerium und Digital Research, für die Informatikausbildung an Bayerns Schulen eingesetzt werden.

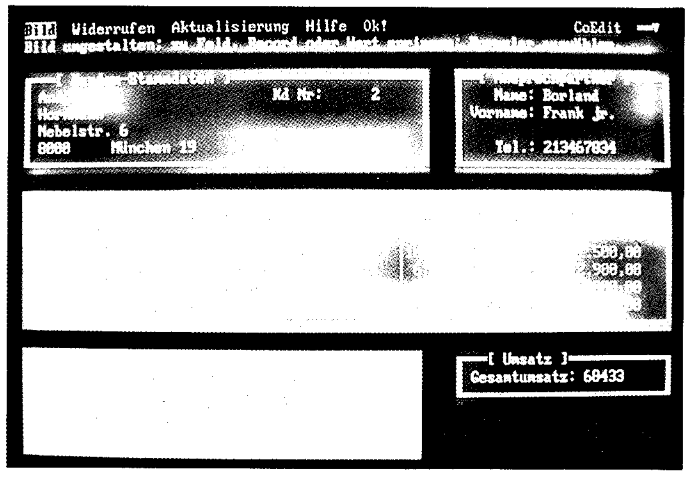
Programmiersprachen, Compiler, Textverarbeitung und Datenbanken

Eine weitere Tendenz in der Softwareentwicklung ist die hin zu immer komplexeren Programmpaketen. So wird einerseits die Grenze zwischen den einzelnen Programmen für die Verwaltung von Datenbanken, zur Textverarbeitung oder zur Tabellenkalkulation, was ihr Leistungsangebot betrifft, immer fließender, und andererseits werden die Unterschiede der Einzelsysteme zu den integrierten Softwarepaketen immer geringer. Die Datenbanksysteme enthalten eigene Texteditoren, die neben den eigenen meist auch die Daten anderer Programme (wie der Tabellenkalkulation) verarbeiten können, und kaum ein Textverarbeitungs- oder Tabellenkalkulationssystem rühmt sich nicht damit, seinerseits eine Datenbank und natürlich einen Texteditor integriert zu haben. Auch vor den Compilerbauern macht diese Entwicklung nicht halt, und so bietet Microsoft seit kurzem eine gemeinsame Entwicklungsumgebung für Assembler- und C-Kode an. Etwas besonderes hat sich auch der nun schon für mehrere Überraschungen bekannte ehemalige Borland-Mitbegründer Niels Jensen – jetzt Präsident der im Januar 1987 von ihm gegründeten Firma Jensen & Partners International (JPI) – einfallen lassen: eine gemeinsame Entwicklungsumgebung für die von JPI entwickelten Topspeed-Compiler. Zunächst besteht sie aus Topspeed-Modula-2, C und Pascal, aber auch Topspeed-C++ (1990) und Topspeed-Ada (1991) sollen integriert werden. Der entscheidende Vorteil dieser Umgebung besteht darin, daß alle Topspeed-Compiler mit dem gleichen Kodegenerator und der gleichen Laufzeitbibliothek arbeiten und extrem optimierten Kode erzeugen. Die Konstanten, die Sprünge und die Schleifen werden optimiert, nicht genutzter Kode wird eliminiert, ebenso mehrfach vorhandene Programmteile. Einmal geschriebener Kode kann in jeder anderen Sprache weiterverwendet werden; in Abhängigkeit vom Inhalt der Quellendatei wird der geeignete Compiler oder Assembler automatisch ausgewählt. Die zur Bearbeitung des Quelltextes erforderlichen Kommandos sind – wie eh und je von Borland-Produkten her bekannt – die wohl nicht totzukriegenden Steuerzeichen aus Wordstar; wenngleich dessen Bedeutung als Textsystem immer mehr schwindet. Eine ausführlichere Vorstellung von JPI Modula-2 finden Sie bereits im Heft 7/90 der MP.

Große Ereignisse werfen bekanntlich ihre Schatten voraus, und so war bereits im März die Beta-Version des für den Sommer 1990 angekündigten Professional Development System (PDS) für den Microsoft-Compiler C 6.0 zu sehen. Microsoft bemüht sich damit ebenfalls um eine einheitliche „Arbeitsplattform" für die gesamte Entwicklungsumgebung. Ob es der Quelltexteditor, der C 6.0- oder andere Compiler und Assembler, ob es die Hilfe-Dateien oder der Debugger Codeview 3.0 oder ob es der professionelle C-Ratgeber ist – alle präsentieren sich unter der einheitlichen Oberfläche der *Programmer's Workbench*. Unter den Microsoft-Betriebssystemen (MS-DOS, OS/2) und den grafischen Oberflächen MS-Windows bzw. Presentation Manager läuft **C 6.0 PDS** ebenfalls und soll somit eine höhere Produktivität in der Programmentwicklung garantieren. Der neue Compiler soll nicht nur eine Kompiliergeschwindigkeit von bis zu 80 000 Zeichen pro Minute erreichen (wenn man auf die Optimierung verzichtet und nur noch die geänderten Teile des Programms kompilieren und linken läßt), der kompaktere Kode soll auch etwa 20 Prozent schneller sein als bei der Version 5.1. Eine ausführliche Vorstellung dieser Entwicklungsumgebung haben wir in einem der nächsten Hefte der MP vorgesehen. Neben diesem Compiler widmet sich Microsoft aber sehr der Weiterentwicklung einer der ältesten Programmiersprachen – Basic; mit 25 Jahren nun doch schon der Sturm-und-Drang-Zeit entwachsen und ebenfalls in einer PDS-Version – hier sogar schon 7.0 – nun tiefer in das professionelle Leben eingestiegen. Zur Arbeit mit dem Vorgänger, QuickBasic 4/4.5 lesen Sie bitte unseren Beitrag in diesem Heft. In **Basic PDS 7.0** ist neben einer Reihe neuer Befehle – unter anderem zur Unterstützung der strukturierten Programmierung – auch eine neue Verwaltungsmethode für den indizierten, sequentiellen Zugriff auf Datenbanken (**I**ndexed **S**equential **A**ccess **M**ethod) integriert. Insbesondere die Programmierung von kommerziellen und kaufmännischen Anwendungen soll damit erleichtert werden, was wohl auch mit dem neuen Currency-Datentyp zum Ausdruck gebracht werden soll.

Im Jahre 1983 gründete der aus Frankreich stammende ehemalige Universitätsdozent für Mathematik, Philippe Kahn, im kalifornischen Scotts Valley die Firma Borland International, die weltweit durch Turbo-Pascal bekannt wurde. Zum ersten Mal wurde leistungsfähige Software ohne Kopierschutz und zudem zu günstigen Preisen angeboten, was letztlich die Verbreitung von Pascal und Prolog begründete. Aber auch nützliche Hilfsprogramme, so das weit verbreitete, speicherresidente Sidekick (Plus), das seine Hilfe innerhalb (fast) jedes Anwenderprogramms auf Tastendruck mit Taschenrechner, Kalender, Modem und Notizblock anbietet und ebenso schnell wieder verschwindet, kommen aus dem Hause Borland. Im Bereich der Tabellenkalkulation bietet Borland mit **Quattro Pro** ebenfalls Erstaunliches (Bild 27). Durch die von Borland entwickelte Softwaretechnologie VROOMM (Virtual-Real-Time-Object-Oriented-Memory-Manager) enthält Quattro Pro sowohl ausgereifte Verknüpfungs- als auch Präsentationsfunktionen, aber auch anspruchsvolle DTP-Fähigkeiten. Auf dem Gebiet der Datenbanksysteme stellt Borland mit **Paradox 3.0** (Bild 28) ein leistungsfähiges Paket zur Verfügung. Hier ragen besonders die Methode der Abfrage durch Bei-





◄ *Bild 27*

Bild 28 ▲

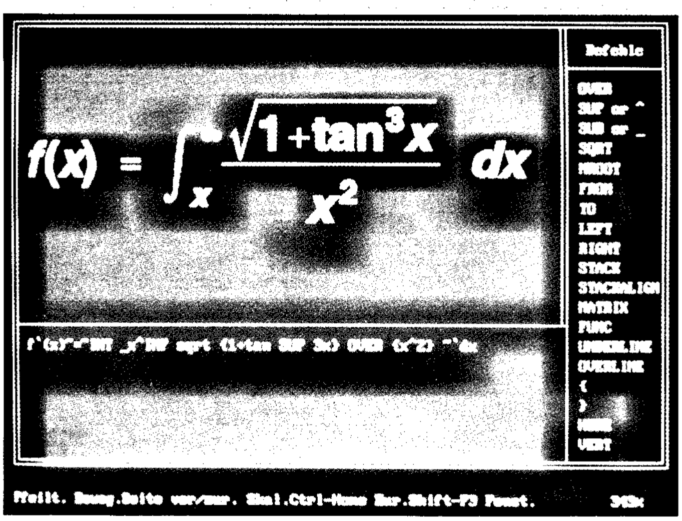
spiel (**Q**uery **B**y **E**xample) und die Funktionen zur Entscheidungsfindung heraus, wodurch es dem Anwender möglich sein soll, auch ohne Programmierkenntnisse Informationen über verschiedene Kombinationsmöglichkeiten erhalten zu können. Die Berichte erreichen dank neuer Farb- und Grafikoptionen DTP-Qualität. Eine interessante Erweiterung für die Borland-Compiler ist sicher **Paradox-Engine**. Sie wird gegenwärtig für C angeboten und soll für Pascal auch ab Sommer 1990 verfügbar sein. Paradox Engine ist eine Bibliothek mit über 70 Funktionen, die Anwendungsprogramme um die Leistungsfähigkeit von Paradox erweitert und so einen unproblematischen Datenbankzugriff ermöglicht.

Die von George Tate und Hal Lashlee 1980 in einer Garage begründete Firma Ashton-Tate konnte leider bisher immer noch nicht die Auslieferung der neuen, fehlerbereinigten Version von dBase IV bekanntgeben, dafür waren aber andere interessante Softwareprodukte im Angebot. So die aktuellen dBase-Versionen in Russisch und Ungarisch. Ein weiteres interessantes Softwarepaket war **Applause II**, mit dem Geschäfts- und Präsentationsgrafiken, aber auch freie Zeichnungen erstellt und sowohl auf Drukker oder Plotter als auch auf Dia-, Folien- oder Filmbelichter ausgegeben werden können.

Ein auf dem Gebiet der professionellen Textverarbeitung interessantes Ereignis war die Vorstellung der Version 5.1 von **Word Perfect**. Über 30 neue Funktionen gegenüber der Version 5.0 steigern die Leistungsfähigkeit. So kann Word Perfect jetzt über Pull-down-Menüs und mit der Maus gesteuert werden. Der integrierte Tabellengenerator ermöglicht es, auch innerhalb der Tabelle Berechnungen vorzunehmen. Für den wissenschaftlichen Satz wird ein Gleichungseditor bereitgestellt, mit dem auch komplizierte Formeln mit mehreren Operanden und mehrstufigen mathematischen Funktionen in den Text integriert werden können (siehe Bild 29). Die kontextbezogene Hilfefunktion, der Druck von 1500 Sonderzeichen, die einfache Erstellung von Etiketten und die Übernahme von Daten aus der Tabellenkalkulation werden als weitere Neuerungen aufgeführt. Word Perfect bietet allerdings einen weiteren entscheidenden Vorteil: Ohne zeitraubendes Umlernen soll es möglich sein, auf verschiedensten Rechnern zu arbeiten. Word Perfect ist verfügbar unter den Betriebssystemen MS-DOS, OS/2 und Unix, kann aber auch auf dem Motorola-Prozessor 68000, auf VAX-Rechnern und dem Großrechner IBM 370 eingesetzt werden. Die deutsche Version ist mittlerweile für etwa 1600 DM netto lieferbar.

Die Software Products International GmbH (SPI) stellte ihr neuestes Produkt, **Open Access III** (Bild 30), vor und gab gleichzeitig bekannt, daß zur Vermarktung von Open Access-Produkten in der DDR mit Robotron eine Vereinbarung abgeschlossen wurde, die jedoch vorerst nur für Open Access II gelten soll. Auf dem deutschen Markt sollen Open Access-Produkte der Marktführer bei integrierten Softwarepaketen sein. Open Access III besteht aus den Modulen Datenbankmanagementsystem, Tabellenkalkulation, Textverarbeitung, Kommunikation, Programmierung, Desk-Manager mit offener Systemarchitektur und Compiler. Die neue Version bietet nun auch zahlreiche neue Funktionen, von denen hier nur einige genannt sein sollen: Open Access III gibt es nicht nur als Einzelplatz- und Netzlösung, auch Client-Server-Konzepte werden unterstützt. Electronic Mail erlaubt den Datenaustausch zwischen den Stationen über das Protokoll X.400, das die Empfangsbereitschaft des Empfängers zum Zeitpunkt des Sendens nicht voraussetzt.

Die Hilfstexte liegen als eigenständige Dateien vor und können damit individuell editiert werden. In C geschriebene Programme und Routinen können, in Open Access III integriert, wie Funktionen genutzt werden. Der Ausdruck von Texten und Grafiken ist sowohl in Postscript als auch in Farbe möglich, wobei die Farben im Ausdruck denen auf dem Bildschirm entsprechen sollen. Daten von dBase II bis dBase IV werden auch mit den als gelöscht markierten Datensätzen gelesen und können zurückgeschrieben werden. Eine binäre Suchstruktur erhöht die Geschwindigkeit von Suchfunktionen. Im Kalkulationsteil ist jetzt auch ein vollwertiges Statistikpaket enthalten. Die Sicherung aller Daten kann automatisiert werden; der Zeitpunkt wird auf den seit der letzten Eingabe verstrichenen Zeitraum bezogen und kann vom Anwender angegeben werden. Die minimalen Hardwarevoraussetzungen sind IBM-kompatible PCs mit einem Hauptspeicher von 384 KByte, einem Diskettenlaufwerk mit 360 KByte Speicherkapazität und 2,5 MByte Festplattenkapazität.



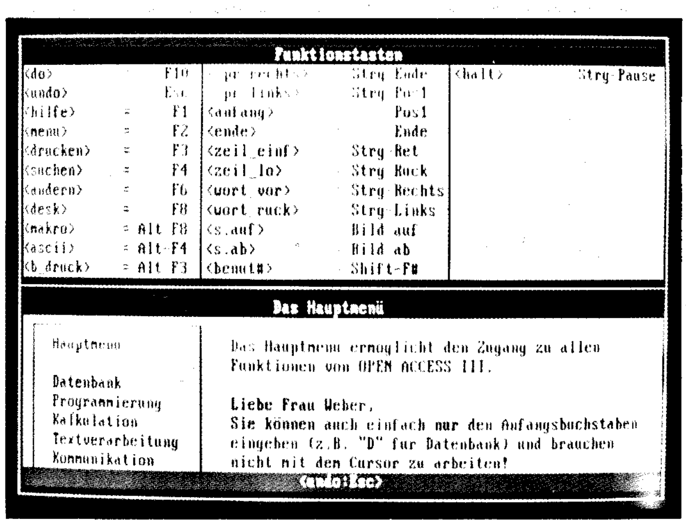


Bild 29

Bild 30

Bitte senden Sie den Coupon an:

Verlag Heinz Heise GmbH & Co KG
Helstorfer Straße 7, 3000 Hannover 61

Zum Verbleib beim Besteller

Ich bestellte am: ______

☐ **c't magazin für computertechnik**
Jahresabonnement 12 Ausgaben
BRD/DDR: DM 91,80;
Ausland: DM 105,60
ab Ausgabe: ______
bis auf Widerruf.
Die Kündigung ist jederzeit mit Wirkung zur jeweils übernächsten Ausgabe möglich.

☐ **iX Multiuser Multitasking Magazin**
Jahresabonnement 6 Ausgaben
BRD/DDR: DM 41,40;
Ausland: DM 47,60
ab Ausgabe: ______
bis auf Widerruf.
Die Kündigung ist jederzeit mit Wirkung zur jeweils übernächsten Ausgabe möglich.

☐ **ELRAD Magazin für Elektronik und technische Rechneranwendungen**
Jahresabonnement 12 Ausgaben
BRD/DDR: DM 71,40;
Ausland: DM 74,60
ab Ausgabe: ______
bis auf Widerruf.
Das Abonnement gilt zunächst für 1 Jahr; es verlängert sich um ein weiteres Jahr, wenn nicht 6 Wochen vor Ablauf des Bezugsjahres schriftlich beim Verlag Heinz Heise gekündigt wird.

☐ **HIFI VISION**
Jahresabonnement 12 Ausgaben
BRD/DDR: DM 78,60;
Ausland: DM 85,80
ab Ausgabe: ______
bis auf Widerruf.
Das Abonnement gilt zunächst für 1 Jahr; es verlängert sich um ein weiteres Jahr, wenn nicht 6 Wochen vor Ablauf des Bezugsjahres schriftlich beim Verlag Heinz Heise gekündigt wird.

Abonnenten haben das Recht, Bestellungen innerhalb von acht Tagen nach Abschluß schriftlich beim Verlag Heinz Heise GmbH & Co. KG, Helstorfer Str. 7, 3000 Hannover 61, zu widerrufen. Zur Wahrung der Frist genügt die rechtzeitige Absendung.

Abo-Bestellcoupon

MP

Ja, übersenden Sie mir bis auf Widerruf alle zukünftigen Ausgaben der angekreuzten Zeitschrift ab Monat: ______

☐ **c't magazin für computertechnik,** Jahresabonnement (12 Ausgaben)
BRD/DDR: DM 91,80; Ausland: DM 105,60
Die Kündigung ist jederzeit mit Wirkung zur jeweils übernächsten Ausgabe möglich.

☐ **iX Multiuser Multitasking Magazin,** Jahresabonnement (6 Ausgaben)
BRD/DDR: DM 41,40; Ausland: DM 47,60
Die Kündigung ist jederzeit mit Wirkung zur jeweils übernächsten Ausgabe möglich.

☐ **ELRAD Magazin für Elektronik und technische Rechneranwendungen,**
Jahresabonnement (12 Ausgaben)
BRD/DDR: DM 71,40; Ausland: DM 74,60
Das Abonnement gilt zunächst für 1 Jahr; es verlängert sich um ein weiteres Jahr, wenn nicht 6 Wochen vor Ablauf des Bezugsjahres schriftlich beim Verlag Heinz Heise gekündigt wird.

☐ **HIFI VISION,** Jahresabonnement (12 Ausgaben)
BRD/DDR: DM 78,60; Ausland: DM 85,80
Das Abonnement gilt zunächst für 1 Jahr; es verlängert sich um ein weiteres Jahr, wenn nicht 6 Wochen vor Ablauf des Bezugsjahres schriftlich beim Verlag Heinz Heise gekündigt wird.

Bitte Rechnung abwarten. (Zahlung ist auf ein Verlagskonto der DDR möglich.)

Vorname/Zuname

Straße/Nr.

PLZ/Wohnort

Datum/Unterschrift

Mir ist bekannt, daß ich diese Bestellung innerhalb von 8 Tagen beim Verlag Heinz Heise GmbH & Co KG, Helstorfer Str. 7, 3000 Hannover 61 widerrufen kann und bestätige dies durch meine Unterschrift. Zur Wahrung der Frist genügt die rechtzeitige Absendung.

Datum/Unterschrift

Bitte beachten Sie, daß diese Bestellung nur dann bearbeitet werden kann, wenn beide Unterschriften eingetragen sind.

ISSN 0232-2892
Mikroprozessortechnik, Berlin 4 (1990) 8
S. 225–256

vorgestellt

Seikosha SL-532

Anzahl der Nadel	24
Druckgeschwindigkeit	
Draft bei 12 cpi bei 10 cpi LQ bei 12 cpi bei 10 cpi	324 cps 270 cps 108 cps 90cps
Grafik	
horizontal vertikal	360 dpi 180 dpi
Fonts	Courier, Prestige Elite
optional	Courier 12, OCR-A, OCR-B, Letter Gothic, Pica, Barcode
Papierhandling	1. Endlospapier über Traktor 2. manuelle Einzelblattzufuhr über Frontseite
optional	3. automatische Einzelblattzufuhr über bis zu 3 Zuführungen
Emulationen	Epson ESC/P, Diabolo 630
optional	HP-GL, Graphtec-GL, Houston-GL
Druckspeicher	8 KByte
optional	16 oder 64 KByte
Schnittstellen (wahlweise)	Centronics, RS-232
Papiermaße	50×90 bis 381×381 mm^2
Anzahl der Durchschläge	4
Zeichen pro Zeile	
Pica Elite Condensed Pica Condensed Elite	136 163 244 272
Lebensdauer der Farbbänder	
schwarz 4farbig	3 Mio Zeichen 1,5 Mio Zeichen
Maße (B × T × H)	$635 \times 405 \times 245$ mm^3
Gewicht	24 kg
Temperaturbereich	
Druck	5...35 °C
Bereitschaft	−20...60 °C
MTBF	4000 h

Seit einiger Zeit machen Drucker durch ein neues Leistungsmerkmal von sich reden: Sie nennen sich Flachbettdrucker. Skeptiker vermuten hier sicherlich nur ein neues Verkaufsargument. Wer aber nicht nur ab und an mal einen Brief schreibt, sondern ständig verschiedene Vorlagen oder sehr unterschiedliche Formate bedrucken muß, der lernt den Komfort von Flachbettdruckern schnell schätzen. Die wesentliche Eigenschaft dieser Drucker ist ihr flaches Druckbett – sie brauchen die Druckvorlage nicht um die Walze herumführen, sondern schieben die Vorlage flach unter dem Druckkopf hindurch. Dadurch können problemlos Briefumschläge, Aufkleber und Pappen bedruckt, aber auch Briefe mit mehreren Durchschlägen geschrieben werden. Hierbei paßt sich der Druckkopf automatisch an die Dicke der Vorlage an. Damit erwächst den in vielen Büros neben den PCs noch immer vorhandenen Schreibmaschinen eine echte Konkurrenz.
Ein Vertreter der Familie der Flachbettdrucker, der darüber hinaus weitere komfortable Funktionen besitzt, ist der SL-532 der japanischen Firma Seikosha. Äußerst vielfältig sind bei ihm die Möglichkeiten des Papiertransportes. So können sowohl perforiertes Endlospapier als auch Einzelblätter mit manueller und automatischer Zuführung verwendet werden. Für den Wechsel von Endlospapier zu Einzelblättern braucht lediglich eine Taste betätigt werden, damit der Drucker das Endlospapier in eine Parkposition zurückzieht und das Einzelblatt einzieht. Die Zuführung des Papiers ist sowohl von vorn als auch von hinten möglich, wobei der Drukker automatisch in der richtigen Richtung druckt. Und herausgeschoben werden kann das Papier nach vorn, nach oben und nach hinten – je nach Auswahl und nach Papierzuführung. Für den Transport kann per DIP-Schalter eine Vorzugsrichtung festgelegt werden. Stellt der Drucker aber bei der automatischen Dickenmessung fest, daß eine zu dicke Vorlage für einen Transport nach oben gebogen werden müßte, dann entscheidet er sich selbständig für einen anderen, flachen Transportweg. Die automatische Dickenmessung dauert etwa 10 Sekunden, deshalb wird sie beim Druck mehrerer Seiten hintereinander auch nur einmal ausgeführt.
Die Dicke der Druckvorlagen ist natürlich begrenzt, sie beträgt 0,325 Millimeter. Das entspricht etwa der Dicke einer Schnellhefterpappe; diese zieht er auch mühelos ein. Überschreitet man die Obergrenze, dann zieht der Drucker trotzdem die Vorlage ein und bedruckt sie auch, es kann dann aber vorkommen, daß der Einzug nicht ganz gerade erfolgt und die Zeilen zu tanzen beginnen. Schwierigkeiten bekommt der Drucker auch bei bestimmten Papierfarben: Verwendet man für Durchschläge das umweltfreundliche grüne Aros-Papier, dann kann es schon einmal vorkommen, daß der SL-532 bockt. Deshalb sollte laut Dokumentation das Papier eine Lichtreflexion von 80 % aufweisen.
Zum Standard des SL-532 gehört die Umschaltmöglichkeit auf einen stärkeren Anschlag. Auch eine automatische Einzelblattzuführung mit bis zu drei Papierzuführungen ist optional möglich.
Der SL-532 weist eine durchschnittliche Geschwindigkeit auf – er druckt im Draft-Modus 324 Zeichen pro Sekunde bei einer Zeichendichte von 12 Zeichen pro Zoll –, dafür hat er aber dank seiner 24 Nadeln ein gutes Schriftbild und kann mit einem entsprechenden Farbband auch mehrfarbig drucken. Die druckbaren Farben sind schwarz, blau, rot und gelb und die Mischfarben lila, grün und orange, wobei an den Ausdruck großflächiger Farbgrafiken keine zu großen Erwartungen gestellt werden sollten.
Der SL-532 wird wahlweise mit einer Centronics- oder einer RS-232-Schnittstelle ausgerüstet. Sein Druckspeicher kann 8, 16 oder 64 KByte betragen. Außerdem bietet er standardmäßgi zwei Druckeremulationen, Epson ESC/P und Diabolo 630, an. Mit einem Plotteraufrüstsatz können auch HP-GL, Graphtec-GL und Houston-GL emuliert werden. Der SL-532 verfügt über ein Bedienfeld, mit dem leicht zwischen den Schriftarten Draft und LQ sowie zwischen den Zeichendichten 10, 12 und 18 Zeichen pro Zoll umgeschaltet werden kann.
Als Fonts sind über DIP-Schalter lediglich Courier und Prestige Elite wählbar. Weitere Fonts – wie Courier 12, OCR, Letter Gothic und Pica – können jedoch über Fontkarten eingelesen werden. Die Auflösung für den Grafikdruck beträgt horizontal 360 Punkte pro Zoll und vertikal 180 Punkte pro Zoll, gedruckt werden kann im Grafikmodus allerdings nur in einer Richtung.
Für die manuelle Papierzuführung können die unterschiedlichsten Papierformate verwendet werden. Die Formate reichen von 50×90 mm^2 – das liegt zwischen DIN A7 und DIN A8 – bis hin zu 381×381 mm^2 – das ist größer als DIN A3. Eine weitere Besonderheit stellt auch der Barcode-Druck dar. Hierzu wird lediglich eine entsprechende Fontkarte gesteckt. Dann können beispielsweise Aufkleber für bestimmte Produkte gedruckt werden. Das ist sicher für viele Gewerbetreibende eine kostengünstige Anwendung dieses Druckers. Allerdings dürfte das unserem Gerät beiliegende englischsprachige Handbuch etlichen Anwendern unnötige Schwierigkeiten bereiten, zumal die Vielfalt der möglichen Funktionen zu einem häufigen Nachschlagen zwingen könnte. Deshalb wäre es für den potentiellen Käufer ratsam, sich die Funktionen für den künftigen Einsatz von einem Fachhändler eingehend vorführen zu lassen. Dazu sind sicher die autorisierten Händler am besten in der Lage. Für Seikosha-Drucker wäre das beispielsweise die Westberliner Firma soft-tronic, die wir Ihnen in MP 4/90 näher vorgestellt haben. Der SL-532 gehört zwar schon zu den Geräten der oberen Preisklasse, seine Funktionsvielfalt und seine Leistungsparameter sollten den Preis jedoch rechtfertigen.

MP-Hk

ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

MP

Heft 9 · 1990

Mikroprozessortechnik

Verlag Technik Berlin

ISSN 0232-2892

DM 5,–

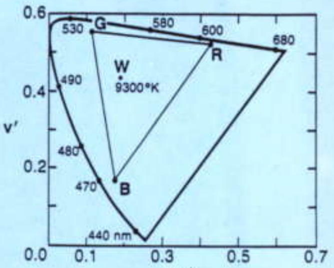
mit Beiträgen aus c't und Computer PERSÖNLICH

Multitasking unter MS-DOS

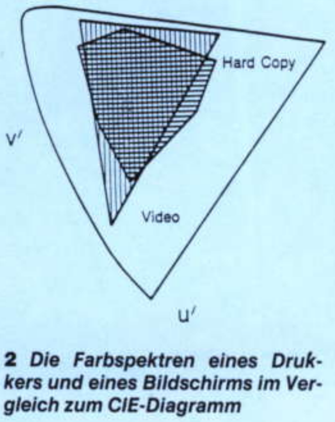
Farbmodell für farbgetreue Darstellung

Wer schon einmal Druckvorlagen oder Muster für Textilien am Computer entworfen hat, kennt das Problem der farbechten Ausgabe des Entwurfs auf einem Drucker oder einem Plotter. Ein Werbeprospekt beispielsweise soll haargenau einen bestimmten Farbton treffen. So darf eine Cola-Büchse nicht schlechthin rot dargestellt werden, sondern es muß das Rot einer ganz bestimmten Firma sein. Aber gerade das war bislang deshalb schwierig, weil die verwendeten Farbmodelle technologieorientiert, also der Hardware angepaßt waren: Für die Ausgabe auf Bildschirm existiert das RGB-System, das den Farben der Schlitzmaske (rot, grün, blau) angepaßt ist. Oder für Farbdrukker wird das CMYK-Farbmodell verwendet, das auf den möglichen Druckfarben cyan, magenta, gelb (**y**ellow) und schwarz (blac**k**) beruht. Beide Systeme verwenden unterschiedliche Prinzipien: Das RGB-Modell verwendet die additive Farbmischung und das CMYK-Modell die subtraktive.

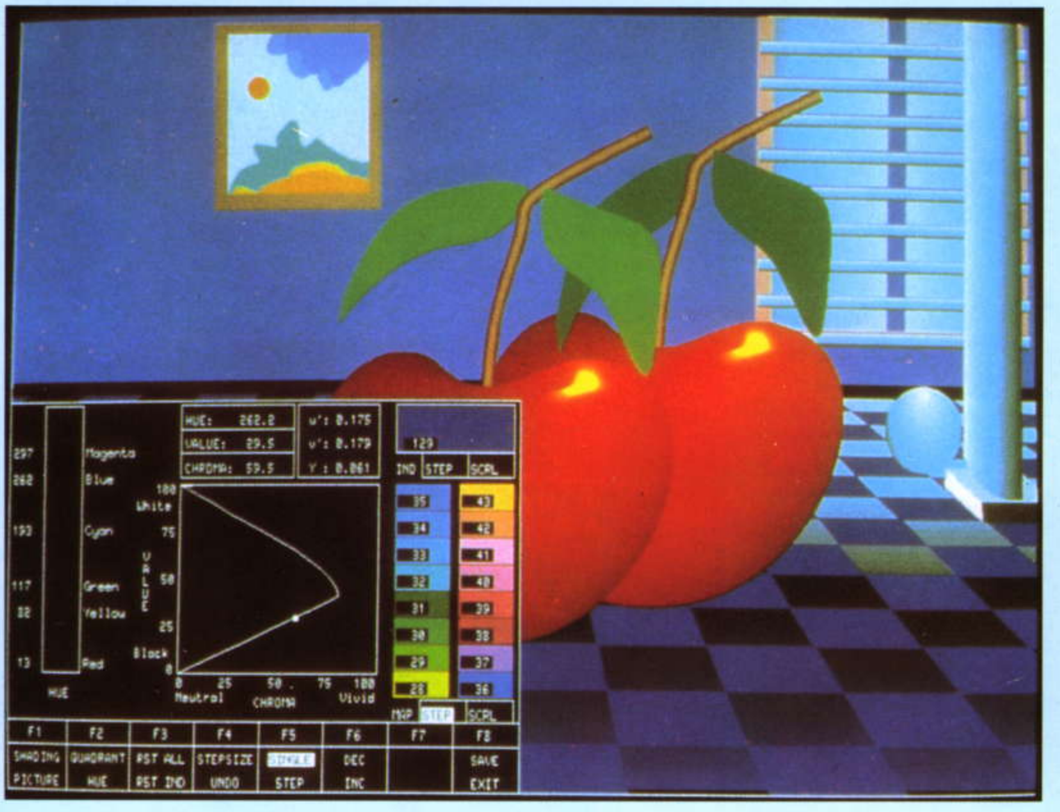
Wie kann man nun aber das auf dem Monitor erzeugte Bild auch farbecht aufs Papier bekommen? Hier hilft ein benutzerorientiertes Farbmodell. Bereits im Jahre 1978 führte die Kölner Firma Tektronix ein solches Modell für ihre Farbgrafikterminals ein – das HLS-Modell. Es hat aber bei all seinen Vorteilen doch noch den Nachteil, daß es die Nichtlinearität der menschlichen Farbwahrnehmung nicht ausreichend berücksichtigt. Deshalb entwickelte Tektronix das neue Farbmodell TekColor, das die



1 Das CIE-Diagramm zeigt die Farbwahrnehmung durch den Menschen; u-Koordinate: Rot-/Grün-Anteil, v-Koordinate: Gelb-/Blau-Anteil



2 Die Farbspektren eines Druckers und eines Bildschirms im Vergleich zum CIE-Diagramm



3 Das interaktive Farbmenü ermöglicht eine Vielfalt der Farbeinstellungen

4 Farbechte Ausgabe auf Bildschirm und Drucker

Eigenschaften der menschlichen Farbwahrnehmung, insbesondere die spektrale Empfindlichkeit des Auges, berücksichtigt. Beispielsweise erscheint dem Auge bei gleicher Intensität die gelbe Farbe heller als die blaue.

Das TekColor-Modell, das auf der diesjährigen Leipziger Frühjahrsmesse vorgestellt wurde, basiert auf dem international anerkannten Farbdiagramm der CIE (Commision Internationale de l' Eclairage) von 1976. Das im Bild 1 dargestellte CIE-Diagramm stellt den Farbraum des menschlichen Auges dar. Die u-Koordinate beschreibt den Rot-/Grün-Anteil, die v-Koordinate den Gelb-/Blau-Anteil und die Vertikale die Helligkeit.

Die drei Komponenten des TekColor-Modells sind Farbe bzw. Schattierung (hue), Helligkeit (value) und Sättigung (chroma), weswegen auch die Bezeichnung HVC-Modell üblich ist. Mit diesem Modell wird es nun möglich, unabhängig von den technologischen Bedingungen auf unterschiedlichen Systemen dieselben Farben zu erzeugen. Allerdings muß dabei berücksichtigt werden, daß die verschiedenen Systeme ein kleineres Farbspektrum als das menschliche Auge aufweisen (siehe Bild 2). Aber auch das Spektrum des Druckers (hardcopy) stimmt nicht völlig mit dem des Bildschirms (video) überein. Deshalb muß der Anwender des HVC-Modells Einschränkungen bei der Farbauswahl hinnehmen, wenn er Farbtreue erreichen will (siehe karierte Fläche in Bild 2). Trotzdem überwiegt sicherlich der Vorteil, daß Farben mit gleicher TekColor-Spezifikation immer übereinstimmen, unabhängig davon, womit sie erzeugt werden. TekColor-Farben lassen sich auch in andere Farbsysteme, zum Beispiel in DIN-Farbstandards, übersetzen.

Damit das HVC-Modell auf dem PC eingesetzt werden kann, bietet Tektronix natürlich auch Werkzeuge an. Ein solches Werkzeug ist das interaktive Farbmenü, das im Bild 3 eingeblendet ist. Hier kann der gewünschte Farbwert (am linken Balken) ausgewählt werden. Helligkeit und Sättigung werden mit dem Value-/Chroma-Diagramm (Mitte) festgelegt. Die dadurch erzeugte Farbe wird im Feld rechts oben angezeigt und kann in das zu erzeugende Bild übernommen werden. Die Einstellungen sind natürlich auch numerisch möglich.

Als erste Anwendung ist das HVC-Modell mit der entsprechenden Farbdruckeranpassung für den Apple MacIntosh II verfügbar (siehe Bild 4). Hier können auch verschiedene Bildschirmtypen angeschlossen werden, die bei der Konfiguration des Rechners anzugeben sind. Schwierigkeiten bereitet noch der Anschluß von Druckern, die neben seriellen und parallelen Schnittstellen auch die RGB-Schnittstelle verwenden. Für die RGB-Schnittstelle werden zwar keine speziellen Schnittstellentreiber benötigt, aber sie ist technologieorientiert. Die Nutzung dieser Schnittstelle wie auch der Standard-PC-Technik sollte eine Aufgabe für künftige Entwicklungen sein. *MP-Hk*

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
4. Jahrgang · ISSN 0232–2892

Mikroprozessortechnik, Heft 9 · 90

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag Verlag Technik GmbH, Oranienburger Str. 13/14, DDR – 1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd, Telefax: 2870259

Geschäftsführer Klaus Hieronimus

Redaktion Hans Weiß, Chefredakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2870203), Hans-Joachim Hill (Tel. 2870209); Sekretariat: Hannelore Kulik (Tel. 2870381)

Gestaltung Christina Bauer

Titel Holger Benicke, Margitta Beyer

Beirat Prof. Dr. sc. Thomas Horn, Prof. Dr. Bernd Junghans, Dr. Dietmar Keller, Prof. Dr. sc. Bernd-Georg Münzer, Prof. Dr. sc. Peter Neubert, Prof. Dr. sc. Dr. Michael Roth (Vorsitzender), Dr. Gerhard Schulze, Prof. Dr. sc. Manfred Seifart, Dr. Dieter Simon, Prof. Dr. sc. Dr. Jürgen Zaremba

Lizenz-Nr. 1710 des Presse- und Informationsdienstes der Regierung der DDR

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

Anzeigenannahme Verlag Technik GmbH, Anzeigenabteilung, Oranienburger Straße 13/14, Berlin, 1020; Tel. 2870291, 2870309, Telefax: 2870254

Anzeigenpreise Preisliste 1 vom 1. Juli 1990

Erfüllungsort und Gerichtsstand Berlin-Mitte. Der Verlag behält sich alle Rechte an den von ihm veröffentlichten Aufsätzen und Abbildungen, auch das der Übersetzung in fremde Sprachen, vor. Auszüge, Referate und Besprechungen sind nur mit voller Quellenangabe zulässig.

Redaktionsschluß 13. Juli 1990

AN (EDV) 49837

Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– DM, Abonnementspreis vierteljährlich 15,– DM; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
Verlag Technik, Abteilung Absatz, PSF 201, Berlin, 1020 oder: *DDR:* sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propagandit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSFR:* PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *Republik Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Republik Ungarn:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nôi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Buchhandel oder Verlag Technik GmbH

Das Standardbetriebssystem für PCs, MS-DOS, ist sowohl bis zur neuesten, verfügbaren Version 4.01 als auch in den angekündigten Versionen 5.0 bzw. 3.5 nur als Einzelplatzsystem konzipiert. Auch nutzt es die Möglichkeiten neuer Prozessoren zu paralleler Abarbeitung mehrerer Programme (Multitasking) nicht. Verschiedene Softwarefirmen bieten nun komplette Betriebssystemerweiterungen an, die das Multitasking, die Arbeit an mehreren Arbeitsplätzen oder sogar beides gestatten. Einige dieser Systeme stellen wir Ihnen in unserem Beitrag auf der **Seite 2** etwas ausführlicher vor.

Störungen der Rechentechnik durch **Computerviren** sind seit mehreren Jahren auch bei uns aufgetreten und häufen sich in der letzten Zeit, insbesondere auch infolge des zunehmend unkontrollierten Datenaustausches. Welche Gefahren von Viren und Störprogrammen ausgehen und welche Schutzmöglichkeiten es ihnen gegenüber gibt, erfahren Sie in unserem Beitrag auf der **Seite 11**.

Wollen Sie sich auch selbständig machen und beispielsweise als Ein-Mann-GmbH & Co. KG den Computer- oder Softwaremarkt bereichern? Dann sollten Sie unbedingt unseren Beitrag „Gründungsfieber" auf der **Seite 31** lesen, um aus berufener Feder viele wertvolle Hinweise für Ihre Geschäftstätigkeit unter marktwirtschaftlichen Bedingungen zu erhalten.

Inhalt

Technik international **2. US**
Farbmodell für farbgetreue Darstellung

Uwe Schulze
Multitasking-Betriebssysteme für den 80x86 **2**

Uwe Hübner
Prozeßkommunikation in UNIX-kompatiblen Betriebssystemen **5**

Bernd Näther
Computerviren **11**

Konrad Schultz
Grundlagen des logischen Programmierens **15**

MP-Kurs **19**
Ulrich Kriegel
Lisp (Teil 2)

Wegbereiter der Informatik
Norbert Wiener **25**

Dirk Blavius, Andreas Senst
Die Hardwareinterrupts beim A 7150 **26**

Peter Wollschlaeger
Fenster schaffen besseren Durchblick **28**

Dieter Butscher
Gründungsfieber (Teil 1) **31**

MP-Literatur **35**

Entwicklungen und Tendenzen **36**

MP-Computer-Club **38**
Kay Hradilak
Compilerkompatible Quelltexte in Turbo Pascal

MP-Bericht **40**
COM '90
Hannover Messe Industrie '90
MP auf Bildungsreise

kennengelernt **3. US**
Atari ABC 286-30

Vorschau
Für das Heft 10/1990 bereiten wir für Sie unter anderem folgende Beiträge vor:
- Lokale Netze
- Standardisierte Betriebssysteminterfaces
- Screen-Safer
- Datenbanken im Vergleich

Multitasking-Betriebssysteme für den 80x86

Uwe Schulze, Berlin

MS-DOS ist zwar das mit Abstand meistverkaufte Betriebssystem für PCs – nicht aber das einzige. Insbesondere die Möglichkeiten neuerer Prozessoren werden von anderen Systemen weit besser genutzt.

Als Alternativen zu MS-DOS fallen Ihnen sicher zuerst Unix oder OS/2 ein; OS/2 setzt sich aber entgegen dem Willen von IBM nur zögernd durch. Dafür gibt es vor allem drei Gründe:
① Die Auslieferung erfolgte erst ein Jahr nach der Ankündigung und wichtige Komponenten (Presentation Manager, LAN-Manager) waren noch später oder nur in einer (fehlerhaften) Vorabversion erhältlich.
② Echte OS/2-Software ist rar. Die meisten übernommenen Programme bieten kaum Vorteile gegenüber den DOS-Versionen. In den Fenstern des Presentation Managers liefen lange Zeit nur die Tabellenkalkulation Excel (Microsoft) und WordPerfect 5.0 /1/. Gleiches Bild auch bei den netzwerkfähigen Anwendungen: Hier hat sich Novell einen großen Marktanteil gesichert.
③ OS/2 läßt sich nur auf Geräten der oberen Leistungsklasse effektiv einsetzen. Obwohl auf dem 80286 lauffähig, bringt nur ein 386er die Vorteile wirklich zum Tragen. Schon für die Basisvariante sind 2 MByte Hauptspeicher nötig; die *Extended Edition* mit Presentation Manager fordert 5,5 MByte! In den Zeiten teurer Speicherbausteine war das keine Werbung für ein Betriebssystem.
In diese Marktlücke sind eine Reihe anderer zum Teil recht leistungsfähiger Betriebssysteme gestoßen. Sie bieten Multitasking- und zum Teil auch Multiuserfähigkeiten (siehe Kasten auf Seite 3) und nutzen einen größeren Adreßraum als 1 MByte.
MS-DOS kann die Möglichkeiten der Hardware (Protected Mode des Prozessors, Hauptspeicher oberhalb der 1-MByte-Grenze nur mit XMS-Treiber oder über Extender) nicht effektiv ausnutzen. Grund dafür ist der Versuch, kompatibel zu bleiben und möglichst alle vorhandene Software lauffähig zu halten. Die meisten der neuen Multitasking-Systeme bieten hier Erstaunliches: Mehrere DOS-Programme sind gleichzeitig im Hintergrund oder parallel in mehreren Fenstern lauffähig. So können entweder zeitaufwendige Anwendungen (Berechnungen, Drucken) in den Hintergrund verlegt werden oder aber es werden Anwendungen im Hintergrund einfach *stehengelassen*, beispielsweise ein Ausschnitt einer Textverarbeitung, auf den von Zeit zu Zeit ein Blick geworfen werden soll, während im Vordergrund an etwas anderem gearbeitet wird. Einige Betriebssysteme bieten deshalb auch eine Nutzeroberfläche, die es Hintergrundprozessen erlaubt, in einem Fenster präsent zu sein.

Microsoft Windows

Microsoft-Windows will eigentlich gar nicht so recht zur Überschrift passen, die da „Alternativen zu MS-DOS" heißt – handelt es sich doch um kein neues Betriebssystem, sondern um eine grafische Nutzeroberfläche, die nachträglich auf DOS aufgesetzt wurde, um die Bedienung zu vereinfachen (siehe auch unser Titelbild). Allerdings ist in beschränktem Umfang Multitasking möglich, mehrere Anwendungen können gleichzeitig in verschiedenen Fenstern beobachtet werden, und es existiert eine Programmierschnittstelle zu C /2/.
Um die Vorzüge der 32-Bit-Prozessoren von Intel ausnutzen zu können, wird eine angepaßte Version Windows 386 angeboten, die im Virtual Mode mehrere MS-DOS-Tasks laufen läßt. Windows ordnet sich in IBMs SAA-Konzept (Systems Application Architecture) ein, das einer einheitlichen Darstellung und Bedienung zum Durchbruch verhelfen soll. Auch die Übernahme von Windows-Source-Kode für den Presentation Manager soll möglich sein.
Für den Nutzer zeigt sich SAA in der Mausbedienung (für Windows ein Muß), dem einheitlichen Menüaufbau, den Leuchtbalken und den Sinnbildern (Icons). Microsoft liefert eine Reihe von Utilities mit (Uhr, Taschenrechner, Notizblock), die die Nachbildung der (unaufgeräumten) Schreibtischoberfläche (englisch: Desktop) auf dem Bildschirm demonstrieren sollen. Sie zeigen zumindest, wie man sich die gleichzeitige Abarbeitung von Programmen unter Windows vorzustellen hat.
Die gleichzeitige Ausführung mehrerer Programme erfordert die Teilung aller Ressourcen. Neben der Tastatur und dem Bildschirm müssen auch der Hauptspeicher, der Prozessor und die Schnittstellen vom Betriebssystem verwaltet werden und sind von der exklusiven Nutzung durch ein Programm ausgeschlossen. Windows kennt kein prioritätsgesteuertes Multitasking; alle Anwendungen werden gleichberechtigt behandelt.
Programme, die eigens für Windows geschrieben wurden (Microsoft nennt in seinem Applikationsverzeichnis 94 professionelle Anwendungen), also das Windows Application Program Interface (API) nutzen, sind quasi gleichzeitig in verschiedenen Fenstern lauffähig. Der Start auf Betriebssystemebene (d. h. außerhalb von Windows) ist nur mit einem *Run-time-Modul* möglich.
Für alle anderen (DOS)-Anwendungen benötigt Windows eine sogenannte PIF-Datei (Program Information File), in der der Ressourcenbedarf festgelegt ist. Das betrifft den notwendigen und den erwünschten Hauptspeicherbedarf sowie den exklusiven Zugriff auf Bildschirm, Tastatur und Schnittstellen. Belegt ein Programm keine Systemressourcen exklusiv (das heißt, es hält sich an die BDOS-Schnittstelle), so ist es auch vollständig unter Windows lauffähig und behindert keine anderen gleichzeitig ablaufenden Programme. Beispielsweise kann für *Debug.com* eine PIF-Datei angelegt werden (Windows stellt dafür das Programm *Pifedit* zur Verfügung), die es erlaubt, in einem Fenster zu *debuggen*, während in einem anderen Fenster gerechnet wird oder die Uhr läuft. Desgleichen kann Command.com problemlos in einem Fenster laufen.
Anwendungen, die direkt auf den Bildschirm zugreifen (und dazu gehört jede grafische Ausgabe) können zwar von Windows gestartet werden, laufen aber nur im *Vollbildmodus.* Der gesamte Bildschirm wird zur Verfügung gestellt und es ist kein Multitasking möglich (bei Windows 386 geht sogar das) – die bereits laufenden Anwendungen werden *eingefroren.*
Benötigt eine Anwendung mehr Hauptspeicher als noch frei ist, so lagert Windows sich selbst sowie einen Teil der Windows-Programme bis auf einen kleinen residenten Kern aus. Dieses sogenannte Swapping stellt eine einfache Form virtueller Speicherverwaltung dar. So können mehrere Anwendungen abgearbeitet werden, deren Gesamtgröße die Hauptspeichergröße weit überschreitet.
Es gibt zwei Wege, Windows zum Auslagern von Programmteilen zu veranlassen: Entweder wird als erwünschter Speicherplatz in der PIF-Datei *minus Eins* angegeben, oder die Anwendung läuft explizit im Vollbildmodus (direkten Bildschirmzugriff eintragen).
Statt des Anlegens einer PIF-Datei können Standardwerte auch in der Datei *Win.ini* festgelegt werden. Diese ähnelt der Datei Config.sys und enthält Angaben zur Konfiguration von Windows. Der Eintrag

command.com = 32

Die Intel-Prozessorfamilie

8088	Der Prozessor des ersten IBM PC. Es handelt sich um einen Prozessor mit interner 16-Bit-Architektur und einem externen 8-Bit-Datenbus. Das hatte vor allem den Vorteil, daß eine Reihe von 8-Bit-Peripherieschaltkreisen weiterhin verwendet werden konnten. Der 8088 verwaltet 1 MByte Speicher in Segmenten von 64 KByte.
8086	Dieser Prozessor ist voll befehlskompatibel zum 8088, besitzt aber einen externen 16-Bit-Bus, der die Übertragung beschleunigt.
80186	Verbesserter 16-Bit-Prozessor, der aber kaum Verwendung in PCs fand; man findet ihn auf einigen Netzwerkkarten.
80286	Der Prozessor des IBM PC/AT. Er kennt einen neuen Betriebsmodus (Protected Mode) und kann 16 MByte Hauptspeicher direkt (bzw. 1 Gigabyte virtuell) adressieren.
80386	Ein echter 32-Bit-Prozessor, der einen weiteren Betriebsmodus kennt (Virtual Mode), in dem mehrere 8086-Prozessoren emuliert werden können. Der Prozessor adressiert 4 Gigabyte Hauptspeicher direkt (virtuell 64 Terabyte) und verfügt über eine MMU (Memory Management Unit, deutsch: Speicherverwaltungseinheit). Die Segmentgröße ist auf 4 Gigabyte erweitert.
80386SX	Dieser Prozessor ist voll befehlskompatibel zum 80386, besitzt aber nur einen externen 16-Bit-Datenbus. Er wird besonders attraktiv durch seinen Preis, der nur wenig über dem des 80286 liegt.
80486	Der neuste 32-Bit-Prozessor, der besonders schnell ist, weil ein Teil des Mikrokodes durch Schaltungen ersetzt wurde. Er besitzt bereits einen Arithmetikkoprozessor, eine MMU und einen Cache-Speicher auf dem Chip. In MP 9/89, S. 278 finden Sie eine ausführliche Vorstellung des 80486.

Begriffe

Multitasking: Unter Multitasking wird die scheinbar gleichzeitige Abarbeitung mehrerer Programme (englisch: tasks) verstanden. Echtes Multitasking ist nur auf einem Mehrprozessorsystem möglich, wo jedes Programm von einem eigenen Prozessor bedient wird. Das Betriebssystem übernimmt die Zuteilung von Rechenzeit, Speicherplatz und Zugriffsrechten auf die Peripherie. Auf einem PC (der nur über einen Prozessor verfügt) erfolgt die Zuteilung von Rechenzeit nach dem sogenannten Zeitscheibenverfahren (englisch: time-sharing), bei dem jedes Programm nach einem festen Zeitintervall (wenige Millisekunden) vom Betriebssystem unterbrochen wird und das nächste Programm zu seinem Recht kommt. Für den Anwender entsteht damit der Eindruck der gleichzeitigen Abarbeitung.
MS-DOS ist im Gegensatz dazu ein Singletasking-System. Residente Programme (auch TSR-Programme – Terminate and Stay Resident – genannt) laufen unter MS-DOS nicht im Multitasking-Betrieb, da zu einem Zeitpunkt nur jeweils ein Programm aktiv ist; die anderen Programme werden im Hintergrund *eingefroren*. Dieses Verfahren wird auch *context switching* genannt.

Multiusersystem: In einem Multiusersystem arbeiten mehrere Nutzer (englisch: user) mit eigenen Bediengeräten (Terminals) an einem Computer. Voraussetzung dafür sind Multitasking-Fähigkeiten des Betriebssystems. Je nachdem, ob die Terminals über einen eigenen Prozessor verfügen oder nicht, werden intelligente und nichtintelligente Terminals unterschieden. Multiusersysteme schaffen gute Voraussetzungen für eine Programmentwicklung im Team, da gemeinsam auf alle Ressorcen zugegriffen werden kann. Andererseits ist aber eine leistungsfähige Hardware vonnöten, da sie unter allen Nutzern aufgeteilt wird.

Protected Mode: Die Intel-Prozessoren der 80x86er Serie beherrschen verschiedene Betriebsmodi. Im *Real Mode* – den alle Prozessoren kennen und der aus Kompatibilitätsgründen beibehalten wird – kann nur 1 MByte Hauptspeicher adressiert werden, was die oft zitierte Grenze für MS-DOS bildet. Im **Protected Mode** (deutsch: schützender Modus), der ab dem 80286 zur Verfügung steht, können 16 MByte (80286) bzw. 4 Gigabyte (80386, 80486) adressiert werden, und es bestehen Zugriffsbeschränkungen auf Speicherbereiche der einzelnen Programme. Damit wird ein gegenseitiges Überschreiben verhindert, die Programme also voreinander geschützt (daher der Name). Der Protected Mode ist Voraussetzung für effektives Multitasking. Im **Virtual Mode** – den nur 80386 und 80486 kennen – werden mehrere 8086-Prozessoren emuliert. Er erlaubt damit die gleichzeitige Abarbeitung auch von solchen Programmen, die direkt auf Ressourcen zugreifen, da jedes Programm einen eigenen Interruptvektor und einen eigenen Bildschirmspeicher erhält.
Die Prozessoren starten stets im Real Mode und können softwaremäßig (durch Setzen eines Bits im Steuerregister) umgeschaltet werden. Im Glauben, damit eine höhere Sicherheit zu erreichen, wurde allerdings beim 80286 das softwaremäßige Rückschalten in den Real Mode unterbunden, so daß Programme, die darauf angewiesen sind (z. B. DOS-Extender) zu zeitaufwendigen Tricks greifen müssen. Mit dem 80386 ist diese Beschränkung aufgehoben.

Virtuelle Speicherverwaltung: Bei der virtuellen Speicherverwaltung wird ein größerer Hauptspeicher simuliert, als tatsächlich vorhanden ist, indem das Betriebssystem bei Bedarf Programmteile, die gerade nicht benötigt werden, auf die Festplatte auslagert (Swapping). Diese Technik wird von modernen Prozessoren direkt unterstützt. So kann der 80286 bis zu 1 Gigabyte externen (virtuellen) Speicher auf bis zu 16 Megabyte Hauptspeicher abbilden.

legt einen Speicherbedarf von 32 KByte für den Kommandointerpreter fest und bewirkt, daß dieser auch ohne PIF-Datei gestartet werden kann. Ansonsten verweigert Windows die Ausführung von Standardanwendungen.
Zum Schreiben von Windows-Applikationen liefert Microsoft das *Windows Software Development Kit* /3/. Es besteht aus Bibliotheken für den Microsoft-C-Compiler, die alle I/O-Operationen und Funktionen zur dynamischen Speicherverwaltung ersetzen. Auf diese Weise braucht die gesamte Bedienung der Fensteroberfläche (Vergrößern, Verkleinern und Verschieben von Fenstern, Auswahl von Menüpunkten, etc.) nicht stets neu programmiert zu werden. Multitasking kann entweder durch explizite Übergabe der Steuerung erreicht werden oder über sogenannte *Message Loops*. Es handelt sich dabei um eine spezielle Programmschleife, die auf eine Tastatureingabe wartet, während es Windows gestattet ist, weiter über die Steuerung zu verfügen.
Darüber hinaus werden Editoren für die Icons, die Fonts und die Dialogboxen geliefert, die es erlauben, die Bildschirmgestaltung von Programmen interaktiv zu entwikkeln. Der Ressource-Compiler macht daraus dann linkbare Dateien. Ein spezieller Linker wird mitgeliefert – man benötigt aber noch den Optimierungscompiler von Microsoft für die Übersetzung der eigentlichen Programme. Die Version 3.0 von MS-Windows (Freigabe Mai 1990) wurde sowohl im Aussehen, als auch in der inneren Struktur stark an den Presentation Manager von OS/2 angeglichen. Dazu gehört die Aufteilung in den Programm- und den Dateimanager sowie die integrierte Hilfefunktion. Vorteil an Windows bleibt aber weiterhin, daß schon mit 640 KByte bzw. 1 MByte Hauptspeicher gearbeitet werden kann.
Nach schleppender Marktakzeptanz erlebt MS-Windows inzwischen einen solchen Verkaufsboom, daß auf Drängen von IBM beschlossen worden sein soll, die Entwicklung mit der Version 3.0 abzuschließen, um den Vertrieb von OS/2 nicht zu behindern /4/. Dem widerspricht allerdings die Ankündigung von Windows 4.0 bis Ende 1990 unter Berufung auf Microsoft nahestehende Quellen /5/.

PC-MOS/386

PC-MOS der Firma „The Software Link Inc." ist keine DOS-Erweiterung (wie Windows), sondern ein vollständig neues Betriebssystem. Allerdings ähnelt es MS-DOS im Aufbau stark: Es besteht aus entsprechenden Systemdateien, einem vergleichbaren Kommandointerpreter, kennt die Dateien *Config.sys* und *Autoexec.bat*. Beim flüchtigen Hinsehen wird man kaum bemerken, nicht mit MS-DOS zu arbeiten. Fast sämtliche internen und externen DOS-Kommandos stehen zur Verfügung – zuzüglich eigener Erweiterungen. PC-MOS-Befehle unterscheiden sich von ihrem DOS-Pendant durch einen vorangestellten Punkt /6/.
Das System wird multitasking- und multiuserfähig, indem an jede Task über eine serielle Schnittstelle (bis 115 200 Baud) ein Terminal *angehängt* wird. Das mitgelieferte Programm Emulink versetzt auch Standard-PCs in die Lage, als Terminals zu fungieren. Man kann PC-MOS als 1-, 5- oder 25-Platz-Version erwerben. Aber auch die Single-User-Version ist von Nutzen, da man die Multitaskingfähigkeiten ausnutzen kann. Das spätere Einfügen weiterer Tasks einschließlich des zugeordneten Speicherplatzes, der Portnummer (für den Anschluß von Terminals), der Übertragungsgeschwindigkeit, der Tasknummer und des Startup-Files kann jederzeit mit dem Befehl .Addtask erfolgen.
Auf eine grafische Nutzeroberfläche und die Mausunterstützung wurde verzichtet; alles läuft wie bei DOS im Kommandozeilenmodus. Die Bedienung ist aber (trotzdem oder gerade deshalb?) recht leicht, und beliebte Hilfen wie der Norton Commander können problemlos genutzt werden. Da alle Anwendungen den gesamten Bildschirmspeicher beanspruchen, ist immer nur ein Programm sichtbar – was aber die anderen nicht hindert, fleißig im Hintergrund zu arbeiten. Zum Umschalten des Bildschirms auf ein anderes Programm muß man <Alt> und die Nummer der Task auf dem rechten Ziffernblock drükken. Die Systemtask trägt dabei die Nummer 0. Ein Umschalten zu MS-DOS ist leider nicht möglich. Da es sich bei PC-MOS um ein vollständiges Betriebssystem handelt, hilft nur neu booten von einer anderen Partition oder von der Diskette. Wird die Festplatte für PC-MOS eingerichtet, so kann das erste logische Plattenlaufwerk bis 32 MByte auch von MS-DOS angesprochen werden; für PC-MOS gibt es keine Beschränkungen auf 32 MByte.
Obwohl im Namen enthalten, beschränkt sich die Nutzung von PC-MOS nicht auf 386er Rechner. Allerdings sichert nur er über den Virtual Mode ein Zusammenspiel auch von Programmen, die direkt auf Systemressourcen (z. B. Bildschirmspeicher) zugreifen. Ein zweites Problem ist der Speicherplatz, der reichlich vorhanden sein sollte. So reserviert sich PC-MOS beim Start mit 4 Hintergrundprozessen großzügig etwa 3,5 MByte. Auf einem 286er benötigt PC-MOS spezielle MMU-Karten (Memory Management Unit) um den Extended Memory über 1 MByte adressieren zu können (auch EMS ist möglich)[1]. Für eine Reihe von Standardkarten werden Treiber mitgeliefert.
PC-MOS gestattet eine sequentielle, gepufferte Kommunikation der Prozesse über Pipes. Eine andere Möglichkeit ist der exzellente NetBIOS-Emulator. Ab Version 3.0 können Rechner unter PC-MOS in Novell-Netze eingebunden werden.
Beim gleichzeitigen Lauf von Programmen in verschiedenen Tasks gibt es bei den bekannten Standardanwendungen keine Behinderungen. Wenngleich Anwendungen unter PC-MOS sogar schneller laufen sollen als unter DOS /7/, so habe ich festgestellt, daß sowohl Berechnungen als auch einige Interaktionen (Blättern im Editor von Turbo-Pascal) deutlich langsamer waren.
Einziger Schwachpunkt scheint die Sicherheit bei der Terminalarbeit zu sein. Nicht nur die fehlende Abschirmung beim Dateizugriff ist bedenklich; von der Systemkonsole kann auch in die Programme auf den angeschlossenen Terminals eingegriffen werden. Die neue Version 4.0 soll aber eine entscheidende Verbesserung der Terminalarbeit bringen.

Concurrent DOS

Die Firma Digital Research konnte ihr 16-Bit-

[1] Über die verschiedenen Arten von Speichererweiterungen wird in MP 3/89, S. 89 ausführlich informiert.

Multitasking im MS-DOS-Hintergrund

Auch unter dem Single-Tasking-Betriebssystem MS-DOS 3.xx können beliebige Funktionen in Multitasking-Eigenschaften versetzt werden. Im Bereich Entwicklung wissenschaftlich-technischer Ausrüstungen des Kombinates Carl Zeiss JENA wurde hierzu das Programm DOSFRAME entwikkelt. Es ermöglicht für eine Nutzerfunktion eine komplette Taskumschaltung.
Im installierten Interrupthandler von DOSFRAME ist eine Schnittstelle zum Multiplexinterrupt eingebunden, die eine Kommunikation mit dem Prozeß erlaubt, nachdem er resident installiert wurde. Über diese Schnittstelle können Funktionsanforderungen und Daten an DOSFRAME oder an die eingebundene Nutzerfunktion übermittelt werden.
Der Nutzer wird durch DOSFRAME von aller Systemarbeit bei der Entwicklung von unter Multitasking-Eigenschaften laufenden residenten Programmen entbunden. Bei Verwendung eines 8086-Rechners (5-MHz-Takt) hat DOSFRAME eine Taskumschaltzeit von weniger als sechs Millisekunden. Da für eine Zeitbilanz nur die synchronen Aufrufe von Bedeutung sind, läßt sich auf eine maximale Prozessorbelastung von etwa 10 % schließen. Hier ist die Laufzeit der Nutzerfunktion nicht berücksichtigt. Es werden bei maximaler Generierung 4 KByte Hauptspeicherkapazität benötigt.
Durch die Generierung kann unter anderem eine Wahl der taskumschaltenden Ereignisse erfolgen, die Generierung eines Hotkey zum Starten und Stoppen der Nutzerfunktion veranlaßt oder die beim Start der Nutzerfunktion gültigen Werte für die Aufrufdauer oder die Aufrufhäufigkeit festgelegt werden.
Bei der Erprobung von DOSFRAME wurden unterschiedlichste Nutzerfunktionen realisiert (Wecker, RAM-Diagnose, Directory-Überwachung). Dabei erwies sich DOSFRAME als sehr nutzerfreundlich und flexibel. Da eine Vielzahl von Programmen unter der Inkompatibilität von MS-DOS 4.xx leiden, bietet DOSFRAME den Nutzern von MS-DOS 3.xx die Chance, den Prozeßhintergrund zu erschließen. *MP*

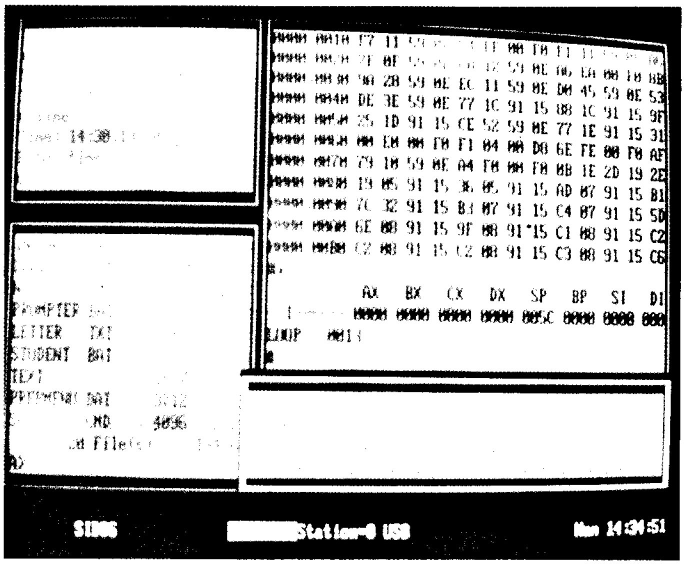


Bild 1 Concurrent DOS XM ist multitasking- und multiuserfähig. Maximal 4 Fenster können in ihrer Größe und auch farblich frei gestaltet werden.

Betriebssystem CP/M-86 (und erst recht Concurrent CP/M) nie gegen MS-DOS durchsetzen, erkannte aber schon recht früh die Mängel der bestehenden Systeme und entwickelte Concurrent DOS (im folgenden kurz CDOS genannt). Heute werden Concurrent DOS XM für 8086- und 80286-Prozessoren und Concurrent DOS 386 für 80386er angeboten. Es handelt sich um ein Multitasking- und Multiusersystem, das zu DOS 3.3 kompatibel ist. Als Terminals (an einer seriellen Multiportkarte) fungieren PCs, von denen in der Grundversion 3 und in der erweiterten Version 6 (CDOS XM) bzw. 10 (CDOS 386) mit einer Übertragungsrate bis zu 38400 Baud angeschlossen werden können.
Wahlweise kann menügeführt (mit Dateimanager FM.CMD) oder auf Kommandoebene gearbeitet werden. Dazu stehen sämtliche von MS-DOS bekannten Befehle (zumeist verbessert oder durch menügeführte Versionen ersetzt), einige CP/M-Utilities (hier kann CDOS seine Herkunft nicht leugnen) sowie 26 zusätzliche Funktionen zur Verfügung. Damit soll etwas mehr Komfort geboten werden, als man es von MS-DOS gewohnt ist. Beispielsweise wird die einfache Belegung der Funktionstasten (auch mit <Alt>, <Ctrl> und <Shift>) durch das Programm *Function.cmd* unterstützt. Ferner ist ein Paßwortschutz für Dateien und Verzeichnissen integriert. Zur Programmentwicklung für die Betriebssystem-Schnittstelle liefert Digital Research den C/CMX-Compiler.
Wird vom Multitasking Gebrauch gemacht, so kann der Window-Manager installiert werden, der es erlaubt, bis zu 4 Anwendungen in verschiedenen Fenstern zu beobachten. Wie in den PIF-Dateien von MS-Windows müssen dazu einige Informationen über den Ressourcenbedarf der Programme (hier Programmcharakteristik genannt) zur Verfügung gestellt werden. Jedes Fenster kann auch auf den ganzen Bildschirm ausgedehnt (gezoomt) oder anderweitig in Größe und Position geändert werden. Die Umschaltung zwischen den Tasks erfolgt mit <Ctrl> und <1> bis <4> auf dem rechten Ziffernblock.
Es ist möglich, beliebige Tasks per Kommando abzubrechen und aus dem Speicher zu entfernen, was auch bei residenten Programmen korrekt läuft.
CDOS ist – da auch hier viel Hauptspeicher vonnöten ist – besonders auf Expanded Memory ausgerichtet. Die XM-Version enthält bereits entsprechende Treiber. Zwar ist CDOS auch auf 640 KByte lauffähig, doch stößt man dann sehr schnell an die Speichergrenzen, zumal die Vergabe an die einzelnen Programme nicht dynamisch erfolgt.
Sie können CDOS und MS-DOS (ab 3.3) gemeinsam auf einer Festplatte installieren, beim Booten wird abgefragt, ob CDOS geladen werden soll. Es ist aber auch später noch möglich, CDOS aus MS-DOS heraus (wie ein Programm) mit *Loadsys.com* zu starten. Ebenso wie PC-MOS (eine tabellarische Gegenüberstellung finden Sie in /8/) stellt CDOS eine Alternative zu Unix- oder LAN-Lösungen dar. Die Einbindung in Novell-Netze ist möglich – allerdings vorzugsweise zum Filetransfer, da dafür nur die Vordergrundtask genutzt werden kann. Prioritäten kennt CDOS nicht; es wird mit einem festen Zeitscheibenverfahren (1/60 Sekunde) gearbeitet.
Aus dem Hause Digital Research gibt es auch noch ein anderes (vollständig MS-DOS-kompatibles) Betriebssystem: **DR-DOS.** Es wird im Gegensatz zum Vorbild nicht nur an OEM-Kunden verkauft, sondern kann einzeln (auch ohne Hardware) erworben werden. Aufgrund des günstigen Preises (etwa 150 DM) bieten aber auch immer mehr Hardwarehersteller ihre Produkte mit DR- statt mit MS-DOS an. Mit der Version 5.0 sollen bisher bemängelte Unzulänglichkeiten ausgeräumt sein – und sie besitzt sogar einen nicht unbeträchtlichen Vorteil gegenüber MS-DOS. Fast das gesamte Betriebssystem kann zwischen den Bildschirmspeicher und das ROM-BIOS oder in den Extended Memory ausgelagert werden, so daß der Nutzer den gesamten Speicher bis zum Bildschirmadapter zur freien Verfügung hat.

DESQview

DESQview der Firma *Quarterdeck Office Systems* ähnelt in vielem MS-Windows. Es handelt sich ebenfalls um einen Zusatz für MS-DOS, der multitasking-, nicht aber multiuserfähig ist. Die 386er Version im Virtual Mode arbeitet mit mehreren DOS-Kopien. Es kann Extended und Expanded Memory Verwendung finden, so daß der einzelnen DOS-Anwendung mehr Hauptspeicher zur Verfügung steht als unter einem originalen MS-DOS.
Jede Anwendung kann bis zu 15 MByte Hauptspeicher nutzen; mindestens sollte 1,5 MByte vorhanden sein, da sonst das *Swapping* die Arbeit stark verlangsamt. Durch die Vergabe von Prioritäten kann die Leistung der einzelnen Anwendung beeinflußt werden. Der mitgelieferte Expanded Memory Manager (QEMM) ist aber auch für MS-DOS hilfreich. DESQview arbeitet mit einer Fensteroberfläche; nichts erinnert mehr an die Zeiten von Kommandozeilen. In den Fenstern kann grafisch gearbeitet werden, und auch mit residenten Programmen gibt es keine Probleme. Unterstützt werden ferner Programme, die mit DOS-Extendern für den Protected Mode geschrieben wurden. DESQview ist in alle wichtigen lokalen Netze voll integrierbar, darunter Novell Netware und OS/2-LAN-Manager. Hier ist nicht nur einfacher Filetransfer möglich; die Netzanwendungen vertragen sich auch mit dem Multitasking-Betrieb (z. B. Netzarbeit im Hintergrund). Allerdings ist dabei ein recht ungünstiges Zeitverhalten in Kauf zu nehmen.
Zur Programmentwicklung wird – ähnlich wie bei Windows – ein Application Program Interface (API) angeboten. Es nutzt den Interrupt 15H und ist vor allem für die Verwaltung sämtlicher Ein- und Ausgaben in Warteschlangen zuständig. Neben den C-Bibliotheken gibt es auch noch Schnittstellen für Pascal, Basic, dBase und Clipper. Unterstützt wird die Programmentwicklung ebenfalls durch einen Menügenerator (hier Panel Designer genannt).

VM 386

VM 386 ist ein Betriebssystem von ganz be-

sonderer Qualität. Wie schon der Name (VM = Virtual Machine) sagt, werden im Virtual Mode des Prozessors (ein 80386 ist erforderlich) mehrere virtuelle DOS-Rechner nachgebildet. Sie können einzeln konfiguriert werden (einschließlich der Zuordnung von Peripherie und der Festlegung von Prioritäten) und sind vollständig voneinander abgeschirmt. Diese Abschirmung geht so weit, daß in jeder Task eine andere DOS-Version gefahren werden kann. Konsequenterweise gilt auch ein Warmstart (<Ctrl>-<Alt>-<Del>) nur für die jeweils aktuelle Task. Der Start von VM 386 erfolgt aus einem eigenen Verzeichnis heraus im Kommandomodus von DOS.
VM 386 ist auch multiuserfähig, der Anschluß von Terminals erfolgt aber nicht über die serielle Schnittstelle, sondern mit einem sogenannten Host-Controller. Er ist als Zusatzhardware notwendig – die Terminalarbeit verbessert sich aber entscheidend. So kann auf den als Terminal eingesetzten PCs im Grafikmodus gearbeitet werden. Weiterhin sind auch die Terminals (Nodes genannt) multitaskingfähig. VM 386 benötigt mindestens 2 MByte RAM und unterstützt alle gängigen Grafikstandards. Für die Arbeit im Netzwerk wird das VM 386 NetPak angeboten.

... und OS/2

Trotz der anfangs erwähnten Schwierigkeiten hat OS/2 Chancen, die Nachfolge von MS-DOS anzutreten, da IBM und Microsoft die Marktführer sind.
OS/2 ist multitaskingfähig, beliebig viele OS/2-Anwendungen können gleichzeitig ablaufen, zusätzlich noch ein MS-DOS-Programm in der sogenannten Kompatibilitätsbox. Hier wird ein DOS-Rechner vollständig nachgebildet – einschließlich des Kommandoprozessors Command.com. OS/2 nutzt einen eigenen Kommandointerpreter (Cmd.exe) oder die grafische Nutzeroberfläche, den Presentation Manager. Für das Umschalten zwischen den Programmen ist der Session Manager (<Alt> <ESC>) zuständig.
Auf Mehrbenutzerarbeit hat man für ein Einprozessorsystem bewußt verzichtet, statt dessen setzt man voll auf Vernetzung: Der OS/2-LAN-Manager soll Novell Netware Konkurrenz machen. Zur Beschleunigung der Plattenarbeit bietet OS/2 ab Version 1.2 das neue Dateisystem HPFS (High Performance File System). Es kann alternativ zum alten FAT-Dateisystem (File Allocation Table) installiert werden. Deshalb nennt Microsoft es Installable File System (IFS) und läßt damit den Weg offen, später noch weitere Dateiverwaltungen zu unterstützen und wie Treiber einzubinden. Das HPFS wirkt der starken Fragmentierung entgegen und ist besonders bei großen Dateien bis zu 4mal schneller als das alte FAT-System. Dateinamen sind nicht mehr auf 8 Zeichen beschränkt; später soll auch zwischen Groß- und Kleinschreibung unterschieden werden.
Ein Plus von OS/2 gegenüber den anderen vorgestellten Systemen ist, daß bereits Compiler für alle wichtigen Programmiersprachen zur Verfügung stehen. Sie unterstützen das OS/2 Application Program Interface (API), denn der alte Sammelinterrupt 21H ist im Protected Mode natürlich tabu. Um aber weiterhin die Entwicklung von MS-DOS-Anwendungen zu ermöglichen, wird zusätzlich eine DOS-konforme Schnittstelle angeboten, die Family Application Program Interface (FAPI) heißt.
Und während die anderen Systeme zumeist „nur" ein Multitasking von Programmen bieten, die noch den DOS-Beschränkungen (640 KByte Hauptspeicher) unterliegen, sind bei OS/2 diese Schranken wirklich aufgehoben.

Informationen zu Preisen und Vertriebsbedingungen aller gängigen DOS-ähnlichen Multitasking-Betriebssysteme finden Sie in /9/.

Literatur

/1/ Borchers, D.: Gefräßiger Geselle: IBMs Pesentation Manager und OS/2 Extended Edition. iX Multiuser-Multitasking-Magazin, Hannover (1989) 2, S. 25
/2/ Microsoft Corporation: Microsoft Windows, Benutzerhandbuch. 1987
/3/ Microsoft Corporation: Microsoft Windows, Software Development Kit, Programmer's Learning Guide. 1987
/4/ IBM und Microsoft geraten sich über OS/2-Zukunft in die Haare. Computerwoche, München 16 (1989) 49
/5/ Windows 4.0 Ende 1990? PC-Woche, München 5 (1990) 22, S. 3
/6/ The Software Link, Incorporated: PC-MOS, Modular Operating System, User Guide. 1988
/7/ db: PC-MOS/386 – echtes Multitasking unter MS-DOS. DOS International, München (1988) 8, S. 70
/8/ Zietlow, M: Multitasking-Betriebssysteme im Vergleich. in DOS International, München (1989) 8, S. 39
/9/ Parker, T.: Multitasking operating system enhancement for the 386. Computer Language (1989) 11

Auto-Anwendertreffen

Am 27. Juni 1990 führte die CeTEC in Ost-Berlin ein erstes Treffen für AutoCAD-Anwender aus Ost (und West) durch. CeTEC ist eine Arbeitsgemeinschaft von Ost- und West-Berliner Firmen und Ingenieuren, die seit längerer Zeit auf AutoCAD spezialisiert sind. Leistungen im gesamten Umfeld dieses Programmes werden auf dem in der DDR entstehenden Markt angeboten, eine gemeinsame Firma ist im Entstehen.
Das Interesse an der Veranstaltung war rege. Unter den Teilnehmern waren Vertreter von etwa 65 Firmen und Institutionen der gesamten DDR. Besucher aus West-Berlin und dem Bundesgebiet waren neben dem Leiter der Niederlassung der Firma Autodesk (dem Hersteller von AutoCAD) in Frankfurt am Main und dem Herausgeber einer CAD-Fachzeitschrift überwiegend an Kontakten interessierte Anwender aus mehreren Bereichen. Da dieses Treffen hauptsächlich generelle Informationen vermitteln sollte, wurden Händler bereits vorher gebeten, die an sich erwünschten Selbstdarstellungen auf einen späteren Zeitpunkt zu verschieben.
Fachleute aus allen Bezirken der DDR berichteten über ihre Erfahrungen mit AutoCAD und stellten eigene Entwicklungen auf verschiedenen Anwendungsgebieten vor.
Gemeinsam mit profilierten Anwendergruppen der DDR und mit Autodesk soll für eine im Oktober 1990 geplante nächste Veranstaltung versucht werden, eine vorläufige Liste von in der DDR entwickelten Branchenlösungen und Hilfsmitteln (z. B. ADI-Treiber für DDR-Technologie) zusammenzustellen und die Gründung einer AutoCAD-Anwendergruppe anzuregen, die als allgemeine Basis für den Austausch von Informationen dienen soll.

✉ KONTAKT ☏

PCAII, D. Kittner, Rosenbecker Straße 19, Berlin, 1143; Tel. 331 69 74

Prozeßkommunikation in Unix-kompatiblen Betriebssystemen

Prof. Dr. Uwe Hübner, Chemnitz

Anspruchsvolle Rechneranwendungen sind oft dadurch gekennzeichnet, daß zur Erfüllung einer Aufgabe mehrere Prozesse quasiparallel ablaufen und kooperieren müssen. Beispiele dafür sind unter anderem Datenbanksysteme mit gleichzeitigem Zugriff durch mehrere Benutzer, Transaktionssysteme (Platzbuchung, ...), Prozeßsteuerungs- und Überwachungssysteme.
Die kooperierenden Prozesse können in unterschiedlichen Prozessoren ablaufen oder aber in einem Prozessor, auf dem durch eine Betriebssystemschale ein Mehrprozeßbetrieb organisiert wird. In beiden Fällen sind Mechanismen erforderlich, die Wechselwirkungen verschiedener Prozesse realisieren; dabei muß einerseits ein Zeitbezug bzw. eine definierte Reihenfolge von Aktivitäten verschiedener Prozesse hergestellt werden (Synchronisation) und andererseits müssen Daten aus einem Prozeß in einem anderen Prozeß verfügbar gemacht werden (Informationsübermittlung).
Die Mechanismen für die Lösung dieser Aufgabe in Unix-kompatiblen Betriebssystemen möchte ich Ihnen im folgenden Beitrag vorstellen (nach /1/, /2/).
Die Mechanismen lassen sich in verschiedene Klassen einteilen (gemeinsame Speicherbereiche, ereignisorientierte Konzepte, Botschaften). Für die in den folgenden Abschnitten dargestellten Programmbeispiele wird ein gemeinsamer Modul (Err.c) für Fehlermeldungen und Instrumentierung vorausgesetzt (Bild 1).

Gemeinsame Speicherbereiche

Der Prozessor kann ablaufenden Prozessen zu relativ willkürlichen Zeitpunkten entzogen werden, was unzusammenhängende, inkosistente Daten in den gemeinsamen Speicherbereichen zur Folge haben kann. Um dies zu vermeiden, dürfen mehrere Prozesse nicht gleichzeitig in Programmabschnitte eintreten, in denen auf solche gemeinsamen Speicherbereiche zugegriffen wird (beispielsweise mit den im nächsten Abschnitt beschriebenen Semaphoren). Den gleichen Ef-

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/times.h>

extern int errno, sys_nerr;
extern char * sys_errlist[];
extern long times();

struct tms tb1, tb2; /* Beginn und Ende */
long atime1, atime2;

/* detaillierte Fehlermeldung ausgeben */
void error(msg)
  char * msg;
{
  fprintf(stderr,"error: %s (%d",msg,errno);
  if (errno > 0 && errno < sys_nerr)
    fprintf(stderr, ": %s)\n", sys_errlist[errno]);
  else
    fprintf(stderr, ")\n");
}

/* detaillierte Fehlermeldung ausgeben und Abbruch */
void fatal(msg)
  char * msg;
{
  error(msg); exit(1);
}

/* falsche Argumente beim Programmaufruf */
usage(s)
  char * s;
{
  fprintf(stderr,"usage: %s\n",s); exit(1);
}

/* Beginn einer Zeitmessung, Zeiten registrieren */
void regtime()
{
  if ((atime1=times(&tb1)) == -1) fatal("times");
}

/* Ende einer Zeitmessung, Zeit (user + system) ausgeben */
void prtime(s)
  char * s;
{
  if ((atime2=times(&tb2)) == -1) fatal("times");
  printf("%s: %ld\n", s,
    tb2.tms_utime-tb1.tms_utime+tb2.tms_stime-tb1.tms_utime);
}
```

Bild 1 Fehlermeldungen und Programm-Instrumentierung

```
#include <stdio.h>
extern char * shmadr;
extern int msgsemid, acksemid;

main(argc,argv)
  int argc;
  char * argv[];
{ int cnt;

  if (argc != 2) { fprintf(stderr,
    "usage: htest1 msganzahl\n"); exit(1); }
  cnt = atoi(argv[1]);
  ipcinit();
  regtime();
  while (cnt-- > 0){
    P(acksemid); /* Warten auf Entnahme der lezten Nachricht */
    if (cnt == 0) *shmadr = 1; /* letzte Nachricht */
    else          *shmadr = 0;
    V(msgsemid); /* Signalisieren der Nachricht an Empfaenger */
  }
  prtime("writer");
  ipcend();
  exit(0);
}
```

Bild 2 Sendeprozeß für gemeinsamen Speicher

Bild 4 Gemeinsame Routinen für Speicherbereiche mit Semaphor-Koordination ▶

```
#include <stdio.h>
extern char * shmadr;
extern int msgsemid, acksemid;

main()
{ char c;

  ipcinit();
  regtime();
  V(acksemid);
  do {
    P(msgsemid); /* Warten auf Nachricht von Sender */
    c  = *shmadr; /* Nachricht auswerten (hier nur 1. Byte) */
    V(acksemid); /* Entnahme der Nachricht signalisieren */
  } while (c == 0);
  prtime("reader");
  ipcend();
  ipcrem();
  exit(0);
}
```

Bild 3 Empfangsprozeß für gemeinsamen Speicher

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/sem.h>

#define SHMKEY 2234567
#define SEM1KEY 2234566
#define SEM2KEY 2234565
#define SHMLEN 0x8000

int shmid; /* Identifikator des gemeinsamen Speichersegments */
char * shmadr; /* Adresse des gemeinsamen Segments */
int msgsemid, acksemid; /* Semaphor-Identifikatoren */
extern char * shmat();

/* gemeinsamen Speicher und Semaphoren initialisieren */
ipcinit()
{
  if ((shmid=shmget(SHMKEY,SHMLEN,IPC_CREAT|0666)) == -1)
    fatal("shmget");
  shmadr = shmat(shmid,0,0); /* Segment zuweisen */
  if ((int)shmadr == -1) fatal("shmat");
  if ((msgsemid=semget(SEM1KEY,1,IPC_CREAT|0666)) == -1)
    fatal("semget");
  if ((acksemid=semget(SEM2KEY,1,IPC_CREAT|0666)) == -1)
    fatal("semget");
}

/* Zuordnung des gemeinsamen Speichers wieder aufheben */
ipcend()
{
  if (shmdt(shmadr) == -1) fatal("shmdt");
}

/* Semaphoroperation "P" - pruefen und (evtl.) warten */
void P(sid)
  int sid;
{
  struct sembuf sb;

  sb.sem_op  = -1;
  sb.sem_flg = 0;
  sb.sem_num = 0;
  if (semop(sid, &sb, 1) == -1) fatal("semop");
}

/* Semaphoroperation "V" - freigeben */
void V(sid)
  int sid;
{
  struct sembuf sb;

  sb.sem_op  = 1;
  sb.sem_flg = 0;
  sb.sem_num = 0;
  if (semop(sid, &sb, 1) == -1) fatal("semop");
}

/* Entfernen aller erzeugten IPC-Objekte */
void ipcrem()
{
  if (semctl(msgsemid,IPC_RMID,0) == -1) fatal("semctl");
  if (semctl(acksemid,IPC_RMID,0) == -1) fatal("semctl");
  if (shmctl(shmid,IPC_RMID,0) == -1) fatal("shmctl");
}
```

fekt erreichen Sie, wenn auch mit etwas drastischeren Mitteln, indem Sie während der betreffenden Programmteile die Prozeßumschaltung überhaupt sperren. Dies ist in Anwenderprogrammen aber nur auf Prozessoren ohne System/User-Modusunterscheidung möglich (i8086).

Gemeinsame Bereiche (shared memory) sind erst in System V realisiert; ihre Handhabung erfolgt durch einen Satz spezieller Systemrufe:

Die Umrechnung des Zugriffsschlüssels in ein Speichersegment liefert den Segmentidentifikator oder −1 bei Fehlern.

int shmget (key, nbytes, flags)
- **long key;** zwischen den Benutzern vereinbarter Schlüssel
- **int nbytes;** Segmentgröße
- **int flags;** Berechtigungen bei neuen Segmenten usw.

Das Segment zuordnen (in den eigenen Adreßbereich abbilden); liefert die Segmentadresse oder −1 bei Fehlern.

char ∗ shmat (segit, addr, flags)
- **int segid;** Segmentidentifikator
- **char ∗ addr;** gewünschte Adresse, 0 – beliebige Adresse
- **int flags;**

Die Segmentzuordnung aufheben; liefert 0 bei Erfolg mit −1 bei Fehlern.

int shmdt (addr)
- **char ∗ addr;** Segmentadresse

Steuerfunktionen; liefern 0 bei Erfolg oder −1 bei Fehlern.

int shmctl (segid, cmd, sbuf)
- **int segid;** Segmentidentifikator
- **int cmd;** Kommando, z. B. IPC_RMID – Bereich entfernen
- **struct shmid_ds ∗ sbuf;** Zeiger auf Statuspuffer

Ereignisorientierte Konzepte

In diese Kategorie wurden Konzepte eingeordnet, die nur Synchronisationsaufgaben erfüllen, aber keine Daten übermitteln; dazu gehören Signale und Semaphore.

Signale

Signale sind in allen Versionen vorhanden, es gibt aber Abweichungen der Signaltypen und deren Semantik.

Das Senden eines Signals erfolgt mit dem Systemruf *Kill*:

int kill (pid, sig) liefert 0 bei Erfolg, −1 bei Fehlern
- **int pid;** Prozeßnummer des Empfängers; 0 – an jeden Prozeß der Prozeßgruppe senden
- **int sig;** Signalnummer, Symbole aus *Signal.h* verwenden

```c
#include <stdio.h>
#include <fcntl.h>
char buf[0x8000];

/* Schreibprozess */
writer(wfd, len, cnt)
  int wfd, len, cnt;
{
  regtime();
  while (cnt-- > 0){
    if (cnt == 0) /* letzte Nachricht */ buf[0] = 1;
    if (write(wfd, buf, len) == -1) fatal("write");
  }
  prtime("writer");
}

/* Leseprozess */
reader(rfd, len)
  int rfd, len;
{ int l;

  regtime();
  while ((l=read(rfd, buf, len)) != -1) {
    if (l==0) sleep(1); /* EOF bei Datei */
    if (buf[0] != 0) { prtime("reader"); return;}
  }
  fatal("read");
}

main(argc, argv)
  int argc;
  char * argv[];
{ int rfd, wfd;

  if (argc != 3) usage("dtest msglength msgcount");
  if ((wfd=open("protokoll", O_WRONLY|O_CREAT, 0600))== -1)
    fatal("open");
  switch (fork()){
  case -1:
    fatal("fork");
  case 0: /* child wird Empfaenger */
    if ((rfd=open("protokoll", O_RDONLY))== -1) fatal("open");
    reader(rfd, atoi(argv[1]));
    break;
  default: /* parent wird Sender */
    writer(wfd, atoi(argv[1]), atoi(argv[2]));
  }
  wait((char *) 0);
  unlink("protokoll");
  exit(0);
}
```

Bild 5 Botschaftenkommunikation über eine Datei

```c
#include <stdio.h>
char buf[0x8000];

/* Schreibprozess */
writer(wfd, len, cnt)
  int wfd, len, cnt;
{
  regtime();
  while (cnt-- > 0){
    if (cnt == 0) /* letzte Nachricht */ buf[0] = 1;
    if (write(wfd, buf, len) == -1) fatal("write");
  }
  prtime("writer");
}

/* Leseprozess */
reader(rfd, len)
  int rfd, len;
{
  regtime();
  while ((read(rfd, buf, len)) != -1) {
    if (buf[0] != 0) { prtime("reader"); return;}
  }
  fatal("read");
}

main(argc, argv)
  int argc;
  char * argv[];
{ int pfd[2];

  if (argc != 3) usage("ptest msglength msgcount");
  if (pipe(pfd) == -1) fatal("pipe");
  switch (fork()){
  case -1:
    fatal("fork");
  case 0: /* child wird Empfaenger */
    /* nicht benoetigten Deskriptor wieder freigeben */
    if (close(pfd[1]) == -1) fatal("close");
    reader(pfd[0], atoi(argv[1]));
    break;
  default: /* parent wird Sender */
    if (close(pfd[0]) == -1) fatal("close");
    writer(pfd[1], atoi(argv[1]), atoi(argv[2]));
  }
  wait((char *) 0);
  exit(0);
}
```

Bild 6 Botschaftenkommunikation über eine Pipe

Als Einschränkung ist zu betrachten, daß die Prozeßnummer des Empfängers bekannt sein und der Sender über die entsprechende Berechtigung gegenüber dem Empfänger verfügen muß (gleicher *effektiver* Nutzer oder *Superuser*). Signale werden nicht gepuffert, das heißt dicht aufeinanderfolgende Signale können verlorengehen oder schlimmer: unerwünschte Wirkungen beim Empfänger haben. Nur in den 4BSD-Versionen werden Signale gepuffert.

Die gezielte Reaktion eines Empfängers auf ein bestimmtes Signal wird erreicht, indem dieser mit dem Systemruf *Signal* ein Behandlungsprogramm für das eintreffende Signal vorplant:

Der folgende Systemruf liefert die alte Behandlung oder −1 bei Fehlern.

int (∗ signal(sig,fcn))()

int sig; Signalnummer, Symbole aus Signal.h verwenden

int (∗fcn)(); neue Signalbehandlung

Signale eignen sich aber so, wie sie konkret implementiert sind, nicht gut als *planmäßige* Kommunikationsmechanismen; ihre Domäne ist die Behandlung von *Ausnahmesituationen*.

Semaphore

Semaphore sind ein bekannter Elementarbaustein für die Prozeßkooperation, sie sind in System V verfügbar:

Die Umrechnung des Zugriffsschlüssels in einen Identifikator liefert den Semaphor-Identifikator oder −1 bei Fehlern.

int semget (key, nsems, flags)

long key; zwischen den Benutzern vereinbarter Schlüssel

int nsems; Semaphoranzahl

int flags; Berechtigungen usw.

Die Semaphoroperation liefert den Semaphorenwert vor der Operation oder −1 bei Fehlern.

int semop (sid, ops, nops)

int sid; Semaphor-Identifikator

struct sembuf (∗ops)[]; Zeiger auf Operationsfeld

unsigned nops; Anzahl der Operationen

Die Steuerfunktionen liefern einen kommandoabhängigen Wert oder −1 bei Fehlern.

int semctl (sid, snum, cmd, arg)

int sid; Semaphoren-Identifikator

int snum; Semaphor-Nummer

int cmd; Kommando

char ∗ arg;

Bei *Semop* zeigt *Ops* auf folgende Struktur:

struct sembuf {**short sem_num;** Semaphor-Nummer
short sem_op; P:−1.V:+1...
short sem_flg; Optionen
};

Semaphore können zur Koordinierung des Zugriffs auf einem gemeinsamen Speicherbereich verwendet werden. Als Beispiel ist dazu in Bild 2 ein Sendeprozeß und in Bild 3 ein Empfangsprozeß dargestellt. Bild 4 enthält die von beiden Programmen benötigten Hilfsroutinen.

Botschaften

Das Botschaftenkonzept vereint die Synchronisation und den Datenaustausch; es stellt einen verhältnismäßig einfach und sicher handhabbaren Mechanismus höheren Niveaus dar.

Da man immer bestrebt sein sollte, in einem System nur einen Mechanismus der Prozeßwechselwirkungen zu verwenden, ist dieses Konzept gut geeignet. Das Botschaftenkonzept ist auf unterschiedlicher Basis (Dateien, Pipes, Fifos, Messages, Sockets) realisierbar.

Dateien

Als Träger für Botschaften können *gewöhnliche* Dateien dienen. Der Inhalt der Datei ist dabei die zu übergebende Information; die Synchronisation zwischen Sender und Empfänger wird dadurch hergestellt, daß der Empfänger ein *End of File* (EOF) erkennt, wenn der Sender noch keine neue Botschaft hinterlegt hat.

Zur Demonstration wesentlicher Programmelemente sowie für Durchsatzuntersuchungen dient das Programm *dtest.c* (Bild 5). Die Operation *fork* liefert die beiden kommunizierenden Prozesse. In dem Programm ist der Kode beider Prozesse enthalten, was jedoch nur bei kleinen Programmen sinnvoll ist. Ansonsten wird eine Verschwendung von Hauptspeicherplatz vermieden, indem einer der beiden Prozesse nach *fork* einen neuen Programmkode mit *exec* lädt. Um zu vermeiden, daß der Leseprozeß die Datei bei *Open* als nicht existent vorfindet (und mit Fehlermeldung abbricht), wird die Datei vor der Operation *fork* hergestellt. Eine andere Variante wäre die zyklische Wiederholung des Open-Versuchs durch den Empfänger.

Wenn EOF erkannt wird, muß der Empfänger warten. Dies kann durch eine ständige Wiederholung des Leseversuchs (verursacht hohe Rechnerbelastung) erfolgen, besser aber – wie dargestellt – mit wiederholten Versuchen in gewissen Zeitabständen.

Die Unteilbarkeit einer Lese- oder Schreibaktion wird im allgemeinen nicht garantiert, das heißt der Leseprozeß muß auch mit *Bruchstücken* einer Nachricht rechnen und sie dann zusammensetzen können.

Zur Assoziation von Sende- und Empfangsprozeß muß lediglich der Name der als Kommunikationsobjekt genutzten Datei bekannt sein. Die Protokollierung der Kommunikation in einer Datei kann zwar als Nebeneffekt mitunter nützlich sein; der ständig steigende Dateiumfang ist aber bald lästig. Im praktischen Einsatz sollte deshalb in bestimmten Zeitabständen der Dateiinhalt wieder entfernt werden.

Pipes

Mit diesem Mechanismus können einige Nachteile des Datei-Mechanismus vermie-

```
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>

main(argc, argv)
  int argc;
  char * argv[];
{ int wfd, rfd;

  if (argc != 3) usage("ftest msglength msgcount");
  if (mknod("fifo", S_IFIFO|0666, 0) == -1) fatal("mknod");
  switch (fork()){
  case -1:
    fatal("fork");
  case 0: /* child wird Empfaenger */
    if ((rfd=open("fifo", O_RDONLY))== -1) fatal("open");
    reader(rfd, atoi(argv[1]));
    break;
  default: /* parent wird Sender */
    if ((wfd=open("fifo", O_WRONLY))== -1) fatal("open");
    writer(wfd, atoi(argv[1]), atoi(argv[2]));
  }
  wait((char *) 0);
  unlink("fifo");
  exit(0);
}
```

Bild 7 Botschaftenkommunikation über einen Fifo

den werden. Das Programm Ptest.c (Bild 6) löst die gleiche Aufgabenstellung wie das Programm in Bild 5.

Die wesentliche Einschränkung dieses Verfahrens besteht darin, daß die kommunizierenden Prozesse auseinander hervorgegangen (assoziiert) sein müssen, da nur assoziierte Prozesse über die beiden vom Pipe-Systemruf vergebenen Filedeskriptoren verfügen können. Der Pipe-Mechanismus liefert erst dann EOF, wenn der Sender die Pipe schließt, es sind keine Vorkehrungen für partielle Botschaften notwendig.

Fifos

Mit diesem auch als *named pipe* bezeichneten Element werden die Vorteile des Datei- und des Pipe-Kommunikationskonzepts vereint. Die Benutzungseigenschaften eines einmal eröffneten Fifos (first in – first out) entsprechen denen einer Pipe. Fifos stehen ab System III und auch in WEGA zur Verfügung.

Die Assoziation der Kommunikationspartner wird über einen Eintrag im Filesystem (den Fifonamen) hergestellt. Dieser Eintrag wird mit dem sonst nur für Verzeichnisse und *special files* benutzten Systemruf mknod hergestellt. Anschließend kann der Fifo-Name in einem Open-Ruf (fast) wie eine gewöhnliche Datei behandelt werden.

In Bild 7 ist die neue Prozedur *main* vorgestellt, *reader* und *writer* sind aus Bild 5 zu übernehmen. Der Open-Ruf hat die spezielle Eigenschaft, daß die diesen Ruf ausführenden Prozesse so lange blockiert werden, bis das Schreib- und das Leseende des Fifos eröffnet wurden.

Diese mitunter unangenehme Eigenschaft kann unterdrückt werden, wenn das Flag O_NDELAY bei Open verwendet wird. Ein späteres Read erfolgt dann aber auch nichtblockierend, das heißt ein Empfänger müßte wie beim Datei-Konzept den Fifo zyklisch abfragen. Eine vorteilhafte Alternative dazu folgt weiter unten.

Messages

In System V gibt es einen Message-Mechanismus, der mit speziellen Systemrufen angesprochen wird:

Die Umrechnung des Zugriffsschlüssels in einen Identifikator liefert den Warteschlangenidentifikator oder −1 bei Fehlern.

int msgget (key, flags)
  long key; zwischen den Benutzern vereinbarter Schlüssel
  int flags; Berechtigungen

Eine Message senden; liefert 0 bei Erfolg oder −1 bei Fehlern.

int msgsnd (qid, buf, nbytes, flags)
  int qid; Warteschlangenidentifikator
  struct msgbuf ∗ buf; Zeiger auf Message
  int nbytes; Länge der Message
  int flags;

Eine Message empfangen; liefert Byteanzahl oder −1 bei Fehlern.

int msgrcv (qid, buf, nbytes, mtype, flags)
  int qid; Warteschlangenidentifikator
  struct msgbuf ∗ buf; Zeiger auf Message
  int nbytes; Länge der Message
  long mtype; Message-Type, 0L – Typ beliebig
  int flags;

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

#define MKEY 1234567
char buf[0x8000];

/* Schreibprozess */
writer(wfd,len,cnt)
  int wfd,len,cnt;
{
  regtime();
  while (cnt-- > 0){
    if (cnt == 0) /* letzte Nachricht */ buf[0] = 1;
    if (msgsnd(wfd,buf,len,0) == -1) fatal("msgsnd");
  }
  prtime("writer");
}

/* Leseprozess */
reader(rfd,len)
  int rfd,len;
{ int l;

  regtime();
  while ((l=msgrcv(rfd,buf,len,0L,0)) != -1) {
    if (buf[0] != 0) { prtime("reader"); return;}
  }
  fatal("msgrcv");
}

main(argc,argv)
  int argc;
  char * argv[];
{ int wfd,rfd;

  if (argc != 3) usage("mtest msglength msgcount");
  switch (fork()){
  case -1:
    fatal("fork");
  case 0: /* child wird Empfaenger */
    if((rfd=msgget(MKEY,0666 | IPC_CREAT))== -1) fatal("msgget");
    reader(rfd,atoi(argv[1]));
    break;
  default: /* parent wird Sender */
    if((wfd=msgget(MKEY,0666 | IPC_CREAT))== -1) fatal("msgget");
    writer(wfd,atoi(argv[1]),atoi(argv[2]));
  }
  wait((char *) 0);
  exit(0);
}
```

Bild 8 Botschaftenkommunikation über Messages

```
#include <stdio.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/times.h>

#define SOCSIZ sizeof(struct sockaddr)
struct sockaddr sockad;
char buf[0x4000];
char * sockname = "tsock";

writer(len, cnt)
  int len, cnt;
{ int wfd;

  if ((wfd=socket(PF_UNIX, SOCK_DGRAM, 0)) == -1) fatal("socket");
  regtime();
  while (cnt-- > 0) {
    if (cnt == 0) buf[0] = 1;
    while (sendto(wfd, buf, len, 0, &sockad, SOCSIZ)) == -1) {
      if (errno != ENOENT) fatal("sendto");
      sleep(1); /* reader noch nicht aktiv */
    }
  }
  prtime("writer");
  close(wfd);
}

reader(len)
  int len;
{ int rfd;

  if ((rfd=socket(PF_UNIX,SOCK_DGRAM,0)) == -1) fatal("socket");
  if (bind(rfd, &sockad, SOCSIZ) == -1) fatal("bind");
  regtime();
  for (;;) {
    if (read(rfd, buf, len) == -1) fatal("read");
    if (buf[0] == 1) break;
  }
  prtime("reader");
  close(rfd);
  unlink(sockname);
}

main (argc, argv)
  int argc;
  char *argv[];
{
  if (argc != 3) usage("stest msglength msgcount");
  sockad.sa_family = AF_UNIX;
  strcpy(sockad.sa_data, sockname);
  switch (fork()) {
  case (-1):
    fatal("fork");
  case 0: /* child wird Empfaenger */
    reader(atoi(argv[1]));
    break;
  default: /* parent wird Sender */
    writer(atoi(argv[1]), atoi(argv[2]));
  }
  wait();
  exit(0);
}
```

Bild 9 Botschaftenkommunikation über Sockets

Die Steuerfunktionen liefern 0 bei Erfolg oder −1 bei Fehlern.

int msgctl (qid, cmd, sbuf)
 int qid; Warteschlangenidentifikator
 int cmd; Kommando, z. B. IPC_RMID –Bereich entfernen
 struct msqid_ds * sbuf; Zeiger auf Statuspuffer

Die Angabe Mtype kann vom Empfänger zur Auswahl aus mehreren eingegangenen Messages verwendet werden. Beim Senden muß ein Wert ungleich 0 im Mtype-Feld von *buf* übergeben werden.

Eine Prozeßkommunikation mit gleicher Funktionalität wie in den vorangegangenen Abschnitten ist in Bild 8 dargestellt.

Sockets

Einen anderen Weg ist man in den 4BSD-Versionen gegangen; das Kommunikationsobjekt wird hier als Socket (Steckdose) bezeichnet. Es kann mit speziellen Systemrufen (socket, send, recv, ...) aber auch mit Read/Write manipuliert werden /3/.

Die neue Qualität besteht darin, daß nicht nur Prozesse in einem Rechner miteinander kommunizieren können, sondern auch solche in verschiedenen Rechnern, die auch über nahezu beliebige Datenkommunikationssysteme (LAN, WAN) miteinander verbunden sein können.

Ein Problem ist dabei allerdings die Adressierung des Empfängerprozesses. Wenn sich der Empfängerprozeß auf dem gleichen Rechner befindet, kann bereits ein Dateiname im lokalen Dateisystem zur Identifizierung der Verbindung dienen; bei lokalen oder landesweiten Netzen muß eine Rechner- und eine Prozeßnummer verwendet werden.

Den unterschiedlichen Aufwand beim Adressieren berücksichtigt man durch Adreßfamilien (Domänen):

AF_UNIX Dateinamen im lokalen Dateisystem
AF_INET ARPA-Netz (TCP/IP-Protokolle)

Den unterschiedlichen Forderungen an die Zuverlässigkeit der Informationsübertragung wird durch eine Typenangabe Rechnung getragen:

SOCK_DGRAM	Datagramme ohne Sicherung gegen Verlust
SOCK_STREAM	verbindungsorientierte, gesicherte Übertragung
SOCK_SEQPACKET	wie STREAM, aber Erhaltung der Paketgrenzen
SOCK_RDM	gesicherte Datagrammübertragung
SOCK_RAW	für Protokolltest usw.

Das Beispiel der letzten Abschnitte läßt sich auch mit Sockets programmieren (Bild 9).

Bei dem hier verwendeten Datagramm-Mechanismus können Fehler auftreten, wenn der Sender die Daten so schnell liefert, daß sie vom System nicht mehr gepuffert werden können; in praktischen Anwendungen sind solche Fälle zu berücksichtigen (oder es sind Stream-Sockets zu verwenden).

Der Datendurchsatz

Um Vorstellungen über die Rechnerbelastung und den möglichen Umfang des Informationsflusses mit den verschiedenen vorgestellten Methoden zu erhalten, wurde mit den angegebenen Programmen die von Sende- und Empfangsprozeß in Anspruch genommene Rechenzeit (Summe aus User time und System time) für eine bestimmte übermittelte Nachrichtenmenge bestimmt und in einen Durchsatzwert umgerechnet. Dieser Wert gilt unter der Annahme, daß keine weiteren Prozesse laufen.

Die Untersuchungen erfolgten für eine Nachrichtenmenge von 256 KByte und verschiedene Botschaftenlängen mit dem System P8000/WEGA:

Botschaftenlänge in Byte	Durchsatz in KByte/s Datei	Pipe	Fifo
32	2,5	3,5	3,5
128	16	23	23
512	36	74	75
2048	44	100	100

Erwartungsgemäß ist der Transport kleiner Botschaften mit großem *System-Overhead* verbunden, die Mechanismen Pipe und Fifo weisen keine nennenswerten Unterschiede auf. Messages und Sockets waren in der benutzten WEGA-Version nicht verfügbar.

Anwendungsbeispiel

Eine V.24-Schnittstelle des Rechners wird als Kopplung zu einem externen Prozeß genutzt. Das können beispielsweise Meßsysteme, zu testende Rechnerstrukturen oder ähnliches sein.

Diese Schnittstelle soll nun von verschiedenen Benutzern quasi gleichzeitig verwendet werden. Die Prozeßkoppelschnittstelle und der externe Prozeß können immer nur eine Anforderung entgegennehmen und geben darauf sofort eine Antwort.

Derartige Probleme werden oft unter Benutzung einer Datei als Semaphor für die Zugriffserlaubnis gelöst; ein Beispiel dafür ist die Belegung einer Rechner-Rechner-Leitung durch Uucp/cu. Dies ist mit einer sehr hohen Rechnerbelastung verbunden, wenn die Benutzer häufig wechseln (d. h. Dauer der Belegung durch einen Nutzer < 1 s).

In prozeßorientierten Systemen kann man das dargestellte Problem so lösen, daß man einen Koordinationsprozeß einführt, der allein berechtigt ist, die ihm unterstehende Ressource (hier die Schnittstelle zum externen Prozeß) zu benutzen.

Andere Prozesse müssen sich als *Kunde* (client) an den *Bediener* (Server) wenden, um ihre *Wünsche* ausgeführt zu bekommen. Das Client-Server-Modell ist über den hier dargestellten Anwendungsfall hinaus in vielen Fällen nützlich (Multiuser-Datenbanken, Namensvergabe ...).

Die Client-Prozesse müssen mit dem Server-Prozeß kommunizieren. Bei der Basis P8000/WEGA ist dafür ein auf dem Fifo-Mechanismus aufbauendes Botschaftenkonzept vorteilhaft einsetzbar.

Die einzelnen Client-Prozesse übergeben ihre Anforderungen an den Server-Prozeß in einem Fifo mit dem Namen *Reqfifo*. Da der Systemruf *Write* auf einem Fifo unteilbar ist (d. h. in ihm keine Prozeßumschaltung stattfindet), werden die Anforderungen unterschiedlicher Client-Prozesse korrekt in der Reihenfolge ihres Eintreffens registriert.

Der Server-Prozeß entnimmt immer eine Botschaft aus *Reqfifo* und führt die in ihr geforderte Aktivität aus. Meist wird eine Antwort bzw. Bestätigung an den jeweiligen Kunden zu übertragen sein. Hier ist das die Information vom externen Prozeß.

Für diese Richtung kommt man nicht mit einem einzigen Fifo aus, sondern man braucht für jeden Client-Prozeß einen separaten Fifo. Dieser erhält den Namen *Fifo_XXXXXX*, wobei für XXXXXX die Prozeßnummer des Client-Prozesses eingetragen wird.

Damit der Server-Prozeß seine *Kunden* auseinanderhalten kann, vergibt er *Kundennummern*. Ein neuer Kunde muß sich also erst unter Angabe des Namens seines individuellen Empfangs-Fifos anmelden und erhält als Antwort eine Kundennummer, unter der er sich zukünftig meldet.

Im Server-Prozeß sind die Filedeskriptoren eine knappe Ressource, da ein Prozeß nur eine begrenzte Zahl von Dateien oder Fifos (meist 20) gleichzeitig geöffnet haben kann. *Kunden*, die die Dienstleistung nicht mehr benötigen, müssen sich daher *abmelden*. Der Empfangs-Fifo wird vom Server-Prozeß geschlossen, der entsprechende Filedeskriptor ist wieder frei.

Die beschriebenen Zusammenhänge bei Anmeldung, normaler Diensterbringung und Abmeldung sind in Bild 10 dargestellt.

Im Programm *Client.c* (Bild 11) und *Server.c* (Bild 12) wird auf die Implementierung des Botschaftenkonzepts in *Msg.c* (Bild 13) zurückgegriffen; notwendige Deklarationen zu den Botschaften enthält *msg.h* (Bild 14).

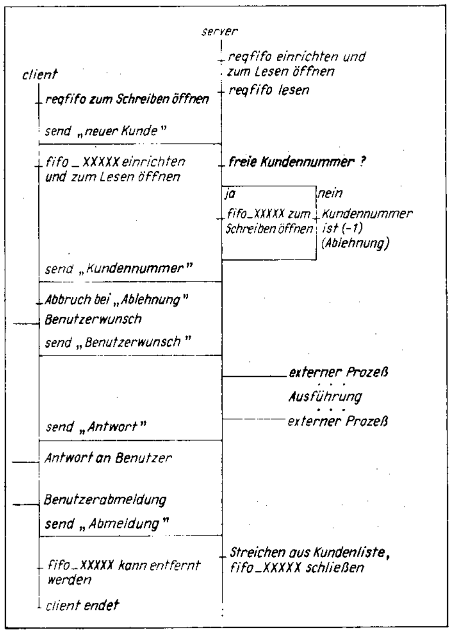


Bild 10 Client-Server-Protokoll

In der Prozedur *Openfifo* wird ein nichtblockierendes *Open* ausgeführt, anschließend aber (für nachfolgende Read-Rufe) wieder der danach sinnvollere, blockierende Modus eingestellt. Bei der Prozedur *Receive* muß trotzdem mit dem Rückgabewert 0 von Read gerechnet werden, nämlich dann, wenn keine Kunden vorhanden sind. Nur in diesem Fall wird zyklisch abgefragt. Zur Übersetzung dieses etwas umfangreicheren Beispiels soll hier noch der Inhalt des verwendeten Makefiles angegeben werden:

```
client: client.o msg.o err.o msg.h
        cc client.o msg.o err.o -o client
server: server.o msg.o err.o msg.h
        cc server.o msg.o err.o -o server
```

Zum Erproben dieses Anwendungsbeispiels ist zuerst der Server im Hintergrund zu starten:

```
%server/dev/tty0 &
```

Anschließend können Client-Prozesse aufgerufen werden. Als Anforderung und Ant-

```
#include <stdio.h>
#include <fcntl.h>
#include <signal.h>
#include "msg.h"

#define CLMAX 3 /* maximale Klientenzahl */

extern int errno;
char * progname;
char * fifoname = "reqfifo";
struct mtype mbuf;
int fdtab[CLMAX];

/* Server beenden, "reqfifo" entfernen */
terminate()
{
  unlink(fifoname); exit(0);
}

sighdl(sig)
  int sig;
{
  errno = 0;
  switch (sig) {
  case SIGHUP:  error("sighup"); break;
  case SIGINT:  error("sigint"); break;
  case SIGPIPE: error("sigpipe"); break;
  default:      error("badsig");
  }
  catchsig();
}

catchsig()
{
  signal(SIGHUP, sighdl);
  signal(SIGINT, sighdl);
  signal(SIGPIPE, sighdl);
  signal(SIGTERM, terminate);
}

/* Aufnahme eines neuen Klienten in die fd-Tabelle */
newclient()
{ int idx;
  int wfd;

  if ((wfd=openfifo(mbuf.mdata, O_WRONLY)) == -1) {
    error("openfifo"); return;
  }
  for (idx = 0; idx < CLMAX; idx++)
    if (fdtab[idx] == -1) {
      fdtab[idx] = wfd; mbuf.index = idx; break;
    }
  mbuf.mlen = 0; /* kein Nachrichtenfeld */
  if (send(wfd, &mbuf) == -1) error("send");
  if (idx == CLMAX) /* kein freier wfd */
    if (close(wfd) == -1) error("close");
}

/* Auftragsausfuehrung fuer einen bekannten Klienten */
oldclient(wfd, fdtty)
  int wfd, fdtty;
{ int le;

  if (mbuf.mlen == 0) { /* Abmeldung */
    if (close(wfd) == -1) error("close");
    fdtab[mbuf.index] = -1;
  }
  else {
    if (write(fdtty, mbuf.mdata, mbuf.mlen) == -1)
      error("write tty");
    if ((le=read(fdtty, mbuf.mdata, DATALEN)) == -1)
      error("read tty");
    mbuf.mlen = le;
    if (send(wfd, &mbuf) == -1) error("send");
  }
}

main(argc, argv)
  int argc;
  char * argv[];
{ int wfd, rfd;
  int fdtty;
  int idx;

  progname = argv[0];
  if (argc != 2) { fprintf(stderr, "usage: %s devname\n",
    progname); exit(1);
  }

  /* alle fd's als unbelegt markieren */
  for (idx = 0; idx < CLMAX; idx++) fdtab[idx] = (-1);

  if ((rfd=openfifo(fifoname, O_RDONLY)) == -1)
    fatal("open reqfifo");
  if ((fdtty=open(argv[1], O_RDWR)) == -1) fatal("open tty");
  catchsig();

  for (;;){
    if (receive(rfd, &mbuf) == -1) {
      error("receive"); continue;
    }
    if (mbuf.index < 0) newclient();
    else {
      if (mbuf.index >= CLMAX || (wfd=fdtab[mbuf.index]) == -1)
        fprintf(stderr, "unknown index: %d\n", mbuf.index);
      else oldclient(wfd, fdtty);
    }
  }
}
```

Bild 11 ***Server-Prozeß***

```
#include <stdio.h>
#include <fcntl.h>
#include "msg.h"

extern char * mktemp();
char * progname;
char * fifoname = "reqfifo";
char * rxname;
struct mtype mbuf;

/* Klient beenden, rxfifo entfernen */
clexit(wfd)
{
  mbuf.mlen = 0; /* Abmeldung */
  if (send(wfd, &mbuf) == -1) error("send");
  sleep(5); /* Unterbrechung des Servers vermeiden */
  unlink(rxname);
  exit(0);
}

main(argc, argv)
  int argc;
  char * argv[];
{ int wfd, rfd;
  int fdtty;
  int le;
  int idx;

  progname = argv[0];
  if (argc != 1) { fprintf(stderr, "usage: %s\n", progname);
     exit(1);
  }
  if ((wfd=openfifo(fifoname, O_WRONLY)) == -1)
    fatal("open reqfifo");
  rxname = mktemp("fifo_XXXXXX");
  strcpy(mbuf.mdata, rxname);
  mbuf.mlen = strlen(rxname) + 1;
  mbuf.index = (-1);
  if (send(wfd, &mbuf) == -1) error("send request");
  if ((rfd=openfifo(rxname, O_RDONLY)) == -1) fatal("open");
  if (receive(rfd, &mbuf) == -1) error("receive");
  idx = mbuf.index;
  if (idx == -1) { fprintf(stderr, "too many users\n"); exit(1);}
  else printf("index: %d\n",idx);

  for (;;){
    /* Warten auf eine Eingabe, "sleep" bei EOF */
    while ((le=read(0, mbuf.mdata, DATALEN)) == 0) sleep(2);
    mbuf.mlen = le; mbuf.index = idx;
    if (mbuf.mdata[0] == 'q') clexit(wfd);
    if (send(wfd, &mbuf) == -1) error("send");
    if (receive(rfd, &mbuf) == -1) error("receive");
    write(1, mbuf.mdata, mbuf.mlen);
  }
}
```

Bild 12
Client-Prozeß ▶

```
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include "msg.h"

#define MAXTRIES 5
#define WTIME 3
extern int errno;

/* Herstellen (wenn notwendig) und oeffnen eines FIFOs */
int openfifo(fname, flags)
  char * fname;
  int flags;
{ int n;
  int fd;

  if (mknod(fname, S_IFIFO|0666, 0) == -1 && errno != EEXIST)
    return(-1);
  for (n = 0; n < MAXTRIES; n++) {
    if ((fd=open(fname, flags | O_NDELAY)) != -1) break;
    if (errno != ENXIO) return(-1);
    sleep(WTIME);
  }
  /* O_NDELAY wieder ausschalten */
  if (fcntl(fd, F_SETFL, flags) == -1) return(-1);
  return(fd);
}

send(wfd, wbuf)
  int wfd;
  struct mtype * wbuf;
{
  return(write(wfd, wbuf, MSGLEN));
}

receive(rfd, rbuf)
  int rfd;
  struct mtype * rbuf;
{ int l;

  for(;;) {
    if ((l=read(rfd, rbuf, MSGLEN)) == -1) return(-1);
    if (l > 0) return(l); else sleep(WTIME);
  }
}
```

Bild 13 ***Botschaften-Modul***

```
#define DATALEN 128
struct mtype { int mlen, index; /* HEAD */
               char mdata[DATALEN];
             };
#define MSGLEN sizeof(struct mtype)
```

Bild 14
Deklarationsdatei

wort dient jeweils eine Textzeile; eine mit q beginnende Textzeile beendet den Client-Prozeß.

An der vom Server bedienten Leitung (hier: /dev/tty0) dürfen keine weiteren Prozesse lesen. Das wird erreicht, indem sie als *Modem-Leitung* konfiguriert wird. Für Experimente kann man auch ein gewöhnliches Login-Terminal benutzen, auf dem ein Deaktivierungsprogramm gestartet wurde:

main() {close(0); pause();}

Die Funktionsweise des hier dargestellten Beispiels wird im Testsystem AGRUS /4/ genutzt, das Erweiterungen bei der Fehlerbehandlung, dem Leitungsprotokoll usw. enthält.

Computerviren

Dr. Bernd Näther, Dresden

Seit 1989 gibt es unter Betreibern und Nutzern der 16-Bit-Rechentechnik unseres Landes zunehmend Verunsicherung im Zusammenhang mit dem Auftreten sogenannter Computerviren.
Diese Verunsicherung resultiert nicht nur aus den damit verbundenen Erscheinungen, sondern auch daraus, daß viele Medien der DDR dieses Thema mehr oder weniger unqualifiziert und nur andeutungsweise behandeln (/1/, /2/) und in der Fachpresse der DDR bisher darüber nur wenig veröffentlicht wurde /3/. Durch Computervirenbefall entstehen erhebliche Risiken für die Nutzung von Rechentechnik, und auch in unserem Land sind erste Schäden durch Viren aufgetreten.

Viren sind eine ganz spezielle Klasse von Störprogrammen, und insofern ist es nicht exakt, immer dann, wenn durchaus verschiedene Störprogramme gemeint sind, den Begriff Virus zu benutzen. So wird fälschlicherweise in /4/ von einem Virus ausgegangen, obwohl es sich um eine sogenannte verschlüsselnde Schadensroutine handelt (in /5/ ausführlich analysiert).
Da der folgende Beitrag der Virenproblematik gewidmet ist, sollen die weiteren Störprogramme lediglich übersichtshalber vorgestellt werden (siehe Bild 1). Die in der Abbildung enthaltenen Fragezeichen mögen andeuten, daß mit weiteren Typen von Störprogrammen durchaus gerechnet werden kann.
Der **Computerwurm** ist ein selbständiges Programm, das sich in einem Rechnernetz ausbreitet, ohne dafür autorisiert zu sein, indem es für das Eindringen in vernetzte Rechner vorhandene Betriebssystemschwachstellen ausnutzt oder sich durch die Analyse und Aneignung der Paßworte anderer Nutzer deren Rechte aneignet.
Ein **Kettenbrief** ist ein ebenfalls in Rechnernetzen auftretendes Störprogramm, das auf dem befallenen Rechner vorhandene *Nutzeranschriften* ermittelt und sich selbst einschließlich der Weiterverbreitungsfunktion an diese Anschriften übermittelt.

Fortsetzung von S. 10

Literatur
/1/ Rochkind, M. A.: Advanced UNIX Programming. Prentice Hall, Englewood Cliffs 1985
/2/ Bach, M. J.: The Design of the UNIX Operating System. Prentice Hall, Englewood Cliffs 1986
/3/ Gauweiler, T.: UNIX als Basis für verteilte Systeme. it 19 (1987) 6, 386–390
/4/ Voigt, W.: ARGUS – flexibles Testsystem für verteilte Steuerungen. Tagung „Computer- und Mikroprozessortechnik 88" Berlin 1988

✉ KONTAKT ☏
Technische Universität Chemnitz, Sektion Informatik, PSF 964, Chemnitz, 9010; Tel. 66 84 64

Das als **Trojanisches Pferd** bezeichnete Störprogramm besitzt zwei wesentliche Bestandteile: Es enthält einerseits für einen Nutzer durchaus nützliche Funktionen, so daß er das von einem fremden Programmierer entwickelte Programm bereitwillig übernimmt, und andererseits eine verdeckte Störroutine. Diese Routine kann – ohne daß dies für den Nutzer sichtbar wird – Störfunktionen am Rechner ausführen (z. B. das Ermitteln von Paßwörtern).
Logische Bomben sind Störprogramme, die erst bei Eintritt eines durch den Programmierer vorgesehenen Ereignisses Störungen verursachen können (z. B. das Löschen aller aktuellen Datenbestände). Derartige Programme sollen als Rache wegen Entlassung durch Programmierer in der Praxis tatsächlich eingesetzt worden sein, um so beispielsweise die Lohnauszahlung der Firma unmöglich zu machen.
Ende 1989 mußte ein weiterer Typ von Störprogrammen – in dem Bild 1 mit **verschlüsselnde Schadensroutine** umschrieben – konstatiert werden. Dieses Programm – unter Vorspiegelung falscher Tatsachen zur Nachnutzung an Forschungseinrichtungen, die sich mit der Bekämpfung des AIDS-Virus beschäftigen, auf Diskette versandt, verschlüsselt in Wirklichkeit die Namen aller in den Verzeichnissen stehenden Dateien, so daß eine Weiterarbeit mit dem Rechner fast unmöglich wird. (Eine etwas ausführlichere Vorstellung dieses speziellen Programms und der Möglichkeiten zur Behebung der entstandenen Schäden finden sie in einem der nächsten Hefte. *MP*)
Um welche Schadensdimensionen es sich beim Auftreten von Störprogrammen handeln kann, soll zumindest am Beispiel eines in den USA im Rechnernetz INTERNET aufgetretenen Computerwurmes verdeutlicht werden: Innerhalb einiger Stunden sollen durch den Computerwurmbefall etwa 6000 Rechner davon betroffen worden sein. Viele Nutzer trennten sich daraufhin teilweise bis zu einer Woche frustriert vom Netz; die Beseitigung des Wurmes soll mehrere Mannjahre hochqualifizierter Sicherheitsexperten und Systemspezialisten gekostet haben.

Was sind nun Computerviren und welche Gefahren gehen von ihnen aus?

Vorerst soll diese Frage nur in sehr allgemeiner Form beantwortet werden: Ein Computervirus ist ein Störprogramm und somit eindeutig ein Softwareprodukt, in der Regel mit der Eigenschaft, durch beabsichtigte (manchmal einschließlich nichtbeabsichtigter) Störeffekte die Nutzung der Rechentechnik zu beeinträchtigen.
Diese Störeffekte können gegebenenfalls zu nichtkorrigierbaren Schädigungen an Programmen und Daten, zur zeitweiligen Nichtverfügbarkeit der Rechentechnik und im Falle von Rechentechnik, die zur Prozeßautomatisierung eingesetzt wird, zu erheblichen Beeinträchtigungen – einschließlich Havarien – führen. So ist beispielsweise ein erster in der DDR aufgetretener Computervirenbefall in einer Prozeßautomatisierungsanlage dank glücklicher Umstände noch glimpflich ausgegangen, eine schwere Havarie wäre aber nicht auszuschließen gewesen. Die Schäden, die durch Computerviren verursacht werden können, sind somit keinesfalls zu vernachlässigen.
Die Idee der Computerviren als sogenannte sich selbst reproduzierende Software wurde von F. Cohen /6/ auf einer Konferenz zu Fragen der Computersicherheit 1984 vorgestellt, nachdem er 1983 erste Experimente mit derartiger Software auf unterschiedlichen Rechnern mit verschiedenartigen Betriebssystemen vorgenommen hatte. Auf Grund der den biologischen Viren ähnelnden Eigenschaft dieser Software, sich selbst zu reproduzieren, prägte Cohen den Begriff Computervirus. Auf welche Weise und von wem dann erstmals in der Praxis Computerviren eingesetzt wurden, läßt sich heute nicht mehr feststellen. Sicher ist jedoch, daß im Gefolge der ab 1984 zunehmenden Publikationen das Interesse für Experimente mit derartiger Software geweckt wurde und wahrscheinlich eine ganze Anzahl heute existierender Viren ihrem Entwickler trotz guter Absicht regelrecht entglitten sind. Beim Softwareaustausch über Ländergrenzen hinweg sind viele Viren in verschiedene Länder (darunter auch in die DDR) regelrecht eingeschleppt worden, wobei ihre „Importeure" davon kaum Kenntnis gehabt haben dürften. Es muß aber auch deutlich gesagt werden, ohne die Motive solcher Leute erhellen zu wollen, daß es Softwarespezialisten gibt und geben wird, die ein wie auch immer geartetes Interesse an der Entwicklung oder Modifikation bzw. Verbreitung solcher Viren haben. Mit diesem Bedrohungspotential werden wir leben müssen – jährlich mehrere hundertausende infizierte Rechner (so wird gegenwärtig eingeschätzt) sind das Ergebnis.

Eigenschaften und Wirkungsweise von Computerviren

Es gibt im Ausgangszustand immer ein Virus-Trägerprogramm (auch Urvirus-Programm genannt), das vom Virusprogrammierer entwickelt und damit Träger und Weiterverbreiter nachfolgender Infektionen anderer Programme wird (Bild 2). Das Virus ist also kein selbständiges Programm, es benötigt in allen Phasen seiner Existenz stets ein Träger- bzw. Wirtsprogramm. Ein Störprogramm wird als Computervirus bezeichnet, wenn es folgende Eigenschaften besitzt:
• Das Störprogramm erkennt durch das am Anfang des Virus stehende Kennzeichenbyte eine bereits erfolgte Infizierung eines Fremdprogramms.
• Es infiziert durch den Viruskern weitere Fremdprogramme, indem der Viruskode in ein noch nicht infiziertes, ausführbares Programm in spezifischer Art und Weise eingebaut wird, wobei alle Eigenschaften des Virus erhalten bleiben.
• Es führt die in der Manipulationsroutine speziellen Manipulationsaufgaben aus.

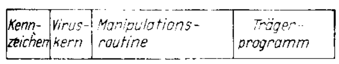
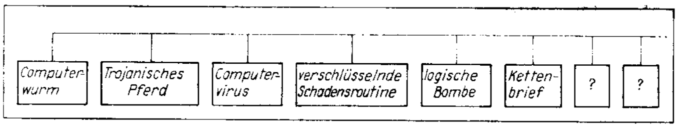


Bild 2



◀ **Bild 1**

Wie verläuft nun eine solche Infektion?

Sie haben beispielsweise die Absicht, ein Spielprogramm von einer ausgeliehenen Diskette auf die Festplatte eines PCs zu übertragen. Dieses Spielprogramm aber ist infiziert. Sekundär dabei ist, ob es sich dabei um die Urversion des Virus oder ein in der Folge infiziertes Programm handelt. Nach dem Kopiervorgang befindet sich das infizierte Programm auf der Festplatte, aber erst nach dem Start des Spielprogrammes – und das ist ein wichtiger Fakt – kann das Virus wirksam werden. Von infizierten, aber nicht gestarteten Programmen geht also vorerst keine Infektion anderer Programme aus – die Gefahr bleibt jedoch latent. Wird also dieses Spielprogramm gestartet, läuft folgender Vorgang ab:
① Das Virus sucht auf der Festplatte und eventuell vorhandenen weiteren Datenträgern nach nichtinfizierten ausführbaren Programmen.
② Es infiziert diese Programme auf die nachfolgend beschriebene Weise.
③ Es kopiert das Programm an seinen Ursprungsplatz zurück. Die Infektion eines Fremdprogrammes kann in überschreibender oder nichtüberschreibender Form erfolgen.
Die überschreibenden Viren überlagern einfach den ersten Teil eines Programmes mit ihrem Kode (Bild 3a), so daß dieses Programm meist nicht mehr lauffähig ist und auch nicht mehr rekonstruiert werden kann. In der Praxis treten derartige Viren zumindest bisher – kaum auf, da sie infolge der gestörten Programmfunktion relativ schnell erkannt werden.
Problematischer und gefährlicher sind nichtüberschreibende Viren, die bei sehr sauberer Programmierung durch keinerlei störende Effekte im Wirtsprogramm auf die Infektion hinweisen, während das nunmehr infizierte Programm bei jedem neuen Start weitere Programme infiziert (Bild 3b).

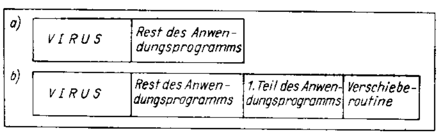


Bild 3

Ein solches *nichtüberschreibendes* Virus verlagert den Programmkode des Fremdprogramms, der nachfolgend durch das Virus überschrieben wird, vorab an dessen Ende. Danach wird der Viruskode in der dargestellten Weise eingebracht. Aufgabe der Verschieberoutine ist es, nachdem das infizierte Programm gestartet wurde und daraufhin das Virus sofort seine Funktionen (Kennzeichen-Prüfung eines weiteren Fremdprogrammes, Infektion, sowie Manipulation) ausgeführt hat, den verschobenen 1. Teil des Programmes zurückverlagern, damit nunmehr das Programm seine Arbeit ordnungsgemäß vollziehen kann. Dieser Virentyp kann somit prinzipiell an einer Verlängerung des Programmkodes des verseuchten Wirtsprogrammes, meist zwischen etwa 600 Byte bis 2 KByte erkannt werden.
Die im eigentlichen Sinne Schaden verursachende Eigenschaft der Viren resultiert aus der spezifischen Funktion der Manipulationsroutine. Welcher Art diese Manipulation ist, hängt von den Absichten des Virenprogrammierers, seinen Systemkenntnissen und dem entsprechenden Betriebssystem ab und kann aus einem schier unerschöpflichen Spektrum möglicher Schädigungen ausgewählt werden, beispielsweise:

- Löschen oder Verändern von Programmen
- Zerstören von Sektoren oder Spuren auf Festplatte oder Disketten
- Neuformatieren von Festplatten oder Disketten und damit Zerstören der darauf befindlichen Daten
- Blockieren aller Eingaben über die Tastatur (sogenannte Crasher-Viren)
- Simulation von (Hardware-)Fehlern.

In /7/ wird sogar von der Möglichkeit der physischen Beschädigung der Hardware gesprochen. Die Manipulation – wie auch die Infektion – kann zudem noch an bestimmte Bedingungen geknüpft sein. Das in der DDR mit Typ 4 bezeichnete Virus (siehe unten) löscht beispielsweise an jedem Freitag, dem 13. – sofern das Jahr ungleich 1987 ist – jedes gestartete Programm.

Viren und Betriebssysteme

Auf dem Gebiet der DDR sind derzeit fast ausschließlich 16-Bit-Rechner unter MS-DOS oder dazu kompatiblen Betriebssystemen befallen worden. Es ist aber ein Trugschluß anzunehmen, daß Rechner mit anderen Betriebssystemen dagegen immun sind. F. Cohen hat bereits 1983 mit IBM/370-Rechnern, mit einer VAX 11/750 und mit weiteren Rechner- bzw. Betriebssystemen die Möglichkeiten eines Virenbefalles auch dieser Rechnertypen hinlänglich demonstriert. Die manchmal vertretene Auffassung, daß Rechner gerade deshalb immun wären, weil durch die betriebssystemgestützte (teilweise auch hardwareunterstützte) Isolation der Nutzer bzw. Aufgaben voreinander eine Infektion fremder Programme prinzipiell ausgeschlossen sei, ist grundsätzlich richtig. Allerdings hat der Computerwurmfall im Rechnernetz INTERNET (bei dem vorwiegend VAX-Rechner mit dem Betriebssystem UNIX betroffen waren) sehr deutlich auf bestehende Sicherheitslücken aufmerksam gemacht, die eine Computervirenattacke ebenfalls durchaus möglich machen.
Warum werden nun vorwiegend MS-DOS-Rechner von Viren befallen?
① MS-DOS besitzt keine wirksamen Mechanismen gegen das Lesen und Ändern fremder Daten, da die vorhandenen Schutzmechanismen (Read-Only- bzw. Hidden-Status) jederzeit verändert werden können.
② MS-DOS bietet dem Virenprogrammierer mit den Möglichkeiten des BIOS (Basic Input Output System) leistungsfähige und hardwarenahe Funktionen zur Beeinflussung des Rechners im gewünschten Sinne und mit der softwareseitigen Interruptorganisation über BIOS- und DOS-Vektorlisten Einflußmöglichkeiten zur Modifikation des Verarbeitungsablaufes.
Besonders die letztgenannte Eigenschaft von MS-DOS ermöglicht eine Variation der Wirkungsweise von Viren in der Form hauptspeicherresidenter Viren.
Im Normalfall läuft beim Aufruf einer Systemfunktion (z. B. der Druckerfunktion) durch ein Anwendungsprogramm unter MS-DOS folgendes ab (Bild 4):

- Der Aufruf erzeugt einen Softwareinterrupt ①.
- Durch die im Interrupt übergebene Interruptnummer erkennt die CPU, von welchem

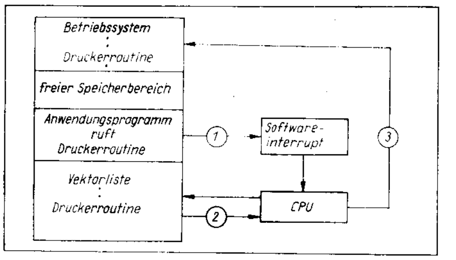


Bild 4

Platz der Vektorliste sie die Adresse der diesen Interrupt behandelnden Routine nehmen mus ②.

- Die Druckroutine wird angesprochen ③ und beginnt zu arbeiten.

Die Rücksprungorganisation zum rufenden Programm ist hier für unsere Belange gegenstandslos.
Die Funktionsweise eines hauptspeicherresidenten Virus soll anhand des Bildes 5 erläutert werden.

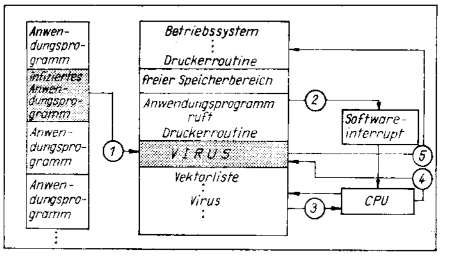


Bild 5

- Das von einem Virus vom Typ nichtüberschreibend und hauptspeicherresident infizierte Programm wird von Festplatte geladen ①; wird das Virus aktiv, verändert es in der Vektorliste einen der Vektoren (hier soll es der Verweis auf die Druckerroutine sein) auf sich selbst; bei Beendigung der Arbeit des infizierten Wirtsprogrammes verbleibt das Virus arbeitsfähig im Hauptspeicher!
- Bei Aufruf der Druckerroutine durch ein beliebiges Programm kommt es ebenfalls zum Softwareinterrupt ②, wobei die CPU jetzt aus der geänderten Vektorliste die Anfangsadresse des Virus übernimmt ③, und damit wird das Virus gestartet ④.
- Erst nach der Abarbeitung des Virus wird die Druckerroutine ausgeführt ⑤, und anschließend erfolgt der Rücksprung in das rufende Programm.

Die Mehrzahl aller nichtüberschreibenden Virentypen ist hauptspeicherresident und „arbeitet" in obiger Weise. Sofern eine oft benötigte Vektorlistenadresse (z. B. die der Druckerroutine) vom Virenprogrammierer verwendet wurde, führt der Start beliebiger Programme mit großer Wahrscheinlichkeit zur Infizierung weiterer Programme.
Im Falle hauptspeicherresidenter Viren existieren somit zwei Gefahrenmomente. Infizierte Programmbestände auf der Festplatte oder der Diskette einerseits und das residente Virus im Hauptspeicher andererseits. Selbst das Neuformatieren der infizierten Datenträger mit anschließendem Neueinspielen von Sicherheitskopien bleibt erfolglos, solange das Virus weiterhin resident ist.
Bei Virenverdacht sollte insofern vor allen anderen Maßnahmen (siehe unten) zuerst durch Reset (besser noch durch kurzzeitiges Ausschalten des Rechners) ein Neuladen des Betriebssystems (möglichst von einer garantiert nicht infizierten Diskette) vorge-

nommen werden. Nur auf diese Weise können hauptspeicherresidente Viren aus dem Hauptspeicher entfernt werden.
Für die Gruppe der Boot-Sektor-Viren stimmt die Aussage leider nicht ganz. Ein Neuladen des Betriebssystems beseitigt diese Viren nur kurzzeitig aus dem Hauptspeicher. Boot-Sektor-Viren befinden sich im Boot-Sektor einer Festplatte (erster Sektor der Platte) bzw. bei Diskette ab Spur 0, Sektor 1, Seite Null. Jedes Neuladen führt damit zur erneuten Infektion. Dieser Virentyp kann nur mit speziellen Antivirenprogrammen (siehe unten) beseitigt werden.

Computervirensituation in der DDR

Der erste Virenfall trat in der DDR im Mai 1988 auf. Nachdem 1988 nur wenige Fälle zu verzeichnen waren, ist seit Sommer 1989 ein rapides Anwachsen sowohl in der Anzahl der Virenfälle als im Hinzukommen neuer Virentypen zu verzeichnen. Im Grunde genommen sagt die Zahl von einigen hundert bis dato aufgetretenen Virenfällen nicht allzuviel aus, zumal auch mit einer erheblichen Dunkelziffer zu rechnen ist. Bedenkenswerter ist es schon eher, daß Virenfälle in nahezu allen volkswirtschaftlichen Bereichen einschließlich des Gesundheitswesens, der Prozeßautomatisierung und von CAD/CAM-Anwendungen aufgetreten sind. Noch hält sich der durch Viren verursachte Schaden in Grenzen, und meist kann die gestörte Hardware wieder in Gang gesetzt werden, wenn auch mit unter Umständen erheblichem Aufwand (Rechenzeitverluste, Arbeitszeitaufwendungen).
Bis Anfang dieses Jahres waren in der DDR etwa 10 verschiedene Virentypen aufgetreten und nach einem fortlaufenden Identifikationsschlüssel benannt worden. Einige spezifische Merkmale dieser Virentypen sind in Tafel 1 enthalten.

Infektionswege

Die Kenntnis über mögliche Infektionswege ist insofern bedeutungsvoll, weil der Nutzer durch angemessenes Verhalten ganz wesentlich dazu beitragen kann, einer Vireninfektion vorzubeugen.
Infektionswege können sein:
① Übernahme fremder Software in einen Kopiervorgang am eigenen oder fremden Rechner
② Nutzung fremder bzw. gemeinsamer Disketten
③ Nutzung eines Rechners ohne vorheriges Abschalten bzw. ohne Reset
④ infizierte Festplatte bzw. Systemdiskette
⑤ infizierte Programmbestände auf einem gemeinsam genutzten File-Server in einem Rechnernetz
Der Praktiker wird sicher feststellen, daß sich hinter vielen im täglichen Umgang durchaus gebräuchlichen Verfahrensweisen die Gefahr verbergen kann, Opfer einer Virenattacke zu werden.
Ein Hinweis zu den Spielprogrammen: Da Spielprogramme oft durch sehr viele Hände gingen, bevor sie beim momentanen Nutzer angelangt sind, ist die Gefahr, daß das Programm irgendwann doch einmal infiziert wurde, besonders groß. Das in vielen Bereichen ausgesprochene Nutzungsverbot für Spielprogramme hat aus diesem Grund schon eine Berechtigung.

Schutz vor Computerviren

Einen allseitig wirksamen und zuverlässig funktionierenden Schutzmechanismus gegen Computerviren gibt es derzeit nicht. Zwar steht mit der Idee objektorientierter Rechnerarchitekturen theoretisch der grundsätzliche Gedanke dafür zur Verfügung, wie ein präventiver Schutz zu gestalten wäre, praktisch wird aber wohl die *Von-Neumann-Maschine* die dominierende Form auch künftiger Rechnergenerationen bleiben. Das in Zusammenhang mit Viren auftretende Problem der Von-Neumann-Maschine ist ihre Gleichbehandlung von Daten und Programmen als mögliche manipulierbare Objekte. Bei objektorientierten Architekturen werden Datenobjekte und Programmobjekte klar voneinander getrennt und damit bezüglich der Manipulierbarkeit streng unterschieden.
Die Möglichkeiten des Schutzes vor Computerviren beschränken sich derzeit auf organisatorische Regelungen und auf den Einsatz programmtechnischer Mittel zur Erkennung und Beseitigung von Viren. Die Nutzer der PCs beugen so durch ein angemessenes Verhalten Infektionen vor. In diesem Zusammenhang sind folgende sechs Aspekte von besonderem Interesse:
① *Aufklärung*
Angemessene Verhaltensweisen in allen Bereichen des gesellschaftlichen Lebens bedürfen primär sachbezogener Kenntnisse, und insofern ist es heute dringend geboten, die Sachkenntnis über Gefahren durch Computerviren und Hinweise zu Schutzmöglichkeiten einem breiten Kreis der Mitarbeiter zugänglich zu machen, um sie dadurch für sachbezogen richtiges Verhalten zu befähigen.
② *Softwarehygiene*
Unter dem Begriff der Softwarehygiene verbirgt sich ein ganzer Bereich technologisch-organisatorischer Maßnahmen mit dem grundsätzlichen Anliegen, Ordnungsmäßigkeit beim Umgang mit Software zu erreichen und bisher übliche – manchmal mit sehr vielen Freiheitsgraden versehene – unangemessene Verhaltensweisen abzulegen. Zu diesem Komplex zählen beispielsweise
• die Autorisierung von Software, das heißt, die in einem bestimmten Bereich einzusetzende Software festzulegen
• die definitive Regelung der Verantwortlichkeit für die Übernahme von Software von betriebsfremden Institutionen. Es wäre zu wünschen, daß ein Zertifikat über die Virenfreiheit von Software generell Bestandteil entsprechender Verträge wird.
• die Erprobung übernommener Software auf Virenbefall auf sogenannten Quarantänerechnern mit rekonstruierbaren Programmen und Daten
• zu beachtende Verhaltensweisen der PC-Nutzer wie Ausschalten des Rechners (sofern er im betriebsbereiten Zustand übernommen wird und das Ausschalten technologisch möglich ist), das Zukleben der Schreibschutzkerne bei den sogenannten Systemdisketten, auf denen sich unveränderbare Programmbestände befinden und die Anwendung der weiter unten aufgeführten Software zur Erkennung eines Virenbefalles.
Eine ganz wesentliche technologische Maßnahme ist natürlich das konsequente Arbeiten mit Sicherungskopien. Es gibt keine Immunität gegen Viren und keine Garantie dafür, daß nicht irgendwann Programm- und Datenbestände unwiederbringlich durch Viren zerstört werden. Existierende Sicherungskopien (die dann hoffentlich nicht auch schon virenverseucht sind) können unter Umständen die einzige Chance für das Wiederherstellen der zerstörten Daten mit relativ geringem Aufwand sein.
③ *Die Software zur Erkennung und Beseitigung eines Virenbefalles* wird in drei Gruppen eingeteilt:
• Virenkontrollprogramme
• Antivirenprogramme
• Befallsüberwachungsprogramme.
Die grundsätzlichen Prinzipien von Virenkontrollprogrammen sind die Verfahren zur Suche nach dem Kennzeichen des Virus in eventuell infizierten Programmen sowie Vergleichsverfahren. Die Suchverfahren basieren auf der Kenntnis der Funktion eines konkreten Virentyps und insbesondere seines Kennzeichenbytes (siehe Bild 2). Jedes damit behaftete Programm wird als von *diesem* Virentyp infiziert betrachtet. Für jeden neu hinzukommenden Virentyp muß jeweils ein weiteres, spezielles Virenkontrollprogramm geschrieben werden. Es gibt mehrere Möglichkeiten, Vergleichsverfahren zur Suche nach infizierten Programmen einzusetzen, so beispielsweise Dummy-File-Verfahren, Verfahren mit Prüfung von Programmparametern sowie kryptografische Verfahren. Da sowohl Dummy-File-Verfahren (Verwendung sogenannter Opferprogramme als Vergleichsindikatoren) als auch kryptografische Verfahren durch den Einsatz der sogenannten Manipulationsgruppenerkennung Spezialfälle sind, soll hier nicht näher auf sie eingegangen werden.
Das Verfahren zum *Suchen von Änderungen* bestimmter Programmparameter (auch Alteration Searcher genannt) ist ausführlich in /9/ beschrieben. Das Wesen dieses Verfahrens beruht darauf, verschiedene, einen aktuellen Programmkode charakterisierende Parameter wie Länge, Checksumme, Attribute, Datum und Uhrzeit der Erstellung und anderes in eine Kontrolldatei zu schreiben. Das Kontrollprogramm ermittelt für das zu kontrollierende Programm die Parameterwerte und prüft anhand der in der Kontrolldatei gespeicherten Werte, ob seit der letzten Prüfung Modifikationen am Programm vorgenommen wurden. Auf diese Weise kann jede Infektion auch durch bisher nicht bekannte Virentypen festgestellt werden; das Prüfprogramm kann darüber hinaus auch zur Erkennung anderweitiger (fahrlässiger oder vorsätzlicher) Manipulationen herangezogen werden.
Die Art und Weise der Einbindung überschreibender Viren in infizierte Programme regte dazu an, sogenannte Antivirenprogramme zu entwickeln, die den erkannten Viruskörper vollständig aus dem Programm entfernen. Eingangs wurde darauf verwiesen, daß ein bestimmter Virustyp ein Programm nicht mehrfach infiziert. Es können aber Viren verschiedenen Typs ein Programm nacheinander infizieren, und dem müssen derartige Antivirenprogramme selbstverständlich Rechnung tragen. Virensuchprogramme, die zum Teil mehr als 100 verschiedene Virentypen erkennen und die entsprechenden Antivirenprogramme dazu werden heute bereits auf dem Softwaremarkt angeboten. Das mit Alteration-Searcher bezeichnete Prüfprogramm (genannt PCC) steht ebenfalls zur Verfügung. Die dynamische Virenbefallssituation auf dem Gebiet der DDR veranlaßt aber zu der Bemerkung, daß anderweitig beschaffte, eventuell nicht so aktuelle Software gegen Viren doch mit einiger Vorsicht betrachtet werden sollte. Das ist keine Kritik an der Qualität derartiger

Tafel 1 Virentypen in der DDR (Stand Januar 1990 in Anlehnung an /8/)

Typ Nr.	auch übliche Bezeichnung	befällt Dateien vom Typ	nicht-über-schreibend	resi-dent	Länge in Byte	Wirkung
1 1a	Wiener Virus	.COM .COM	ja ja	nein nein	648 646	Infizierte Programme können zerstört werden; es kommt zum Systemneustart.
2	Herbst-laub	.COM	ja	ja	1701	Bildschirminhalt wird zerstört, Zeichen fallen auf die unterste Zeile. Virus ist nur im Zeitraum 1. 10. 88–31. 12. 88 aktiv bzw. bei Rechnern ohne Echtzeituhr, wenn kein Datum eingegeben wurde.
3	Italienischer Springer	Boot-Sektor	Boot-Virus			Bootsektoren aller Disketten und der Festplatte werden infiziert, auf dem Bildschirm erscheint ein wandernder Punkt.
4	Israeli-Virus	.COM .EXE	ja	ja	1813 bis 1823	Infiziert alle aufgerufenen Programme, Verarbeitungsgeschwindigkeit des Rechners wird drastisch verringert; zerstört am Freitag, dem 13. (wenn das Jahr ungleich 1987 ist) jedes Programm
5		.COM .EXE	ja	ja	1800 bis 1815	Infiziert alle aufgerufenen Programme (auch beim Kopieren von Programmen!), ein Sektor des aktuellen Laufwerkes wird zerstört.
6		.COM .EXE	ja	ja	bis 1221 bis 1363	Infiziert alle aufgerufenen Programme; Ausgabe eines Tones beim Start eines infizierten Programms.

Drei weitere Viren S1 (Länge 534 Byte), S2 (2875–2890 Byte) und S3 (2946–2961 Byte) infizieren lediglich weitere Programme, ohne besondere Manipulationsaufgaben auszuführen bzw. durch Nebeneffekte Störungen zu verursachen.

Programme, sondern resultiert ganz einfach aus der Überlegung, daß die so beschaffte Software schon einen gewissen Lebenszyklus hinter sich gebracht hat und den aktuellen Erfordernissen damit nicht mehr völlig entspricht.
Das Grundprinzip der Parameterprüfung beim Alteration-Searcher kann neben der obigen Form nutzerveranlaßter Prüfung als Befallsüberwachungsprogramm in geeigneter Weise (z. B. in der Autoexec.bat) in das Betriebssystem eingebunden werden. Von einer solchen Variante wird in /10/ berichtet.

④ Die *Isolation einer Anwendung* ist dann erreichbar, wenn das Eindringen eines Computervirus (über eine Diskette bzw. ein Netz) ausgeschlossen werden kann. Diese Variante ist offenbar nur in ganz speziellen Fällen der Prozeßautomatisierung anwendbar, indem das System, nachdem es vollständig geladen ist, vollständig isoliert wird.

⑤ *Die wirksame Reaktionsfähigkeit*
Im Falle des eingangs geschilderten Computerwurmbefalles im Rechnernetz INTERNET hat sich gezeigt, daß die meisten Nutzer der Situation hilflos gegenüber standen. Ihre Kenntnisse waren nicht ausreichend, wesentliche Gegenmaßnahmen ergreifen zu können. Im Grunde genommen ist die Situation beim Auftreten von Viren sehr ähnlich; einem neuen Typ von Viren sind die meisten Nutzer oder Betreiber der PCs mehr oder weniger ausgeliefert. In Analogie zu den vom USA-Verteidigungsministerium aus dem Vorfall gezogenen Schlußfolgerungen (vgl. /10/) wird künftig auch bei uns eine stärkere Koordinierung und Unterstützung der Nutzer insbesondere mit leistungsfähiger und der aktuellen Virensituation entsprechender Software notwendig werden.

⑥ Es mag etwas ungewöhnlich erscheinen, im Zusammenhang mit Viren *ethisch-moralische Fragen* aufzuwerfen. Straf- und Zivilrecht-Gesetzgebung sind aber nur *ein* notwendiges, keinesfalls jedoch hinreichendes Instrument zur Lösung dieses Problems in der Gesellschaft. Realistisch muß man nämlich einschätzen, daß die Entwickler oder Verbreiter von Computerviren – zumindest mit den heutigen Methoden – in den seltensten Fällen dingfest gemacht werden können. Wie auch für andere Gebiete des menschlichen Zusammenlebens bedarf die Gesellschaft offenbar auch bestimmter ethisch-moralischer Verhaltensnormative für den Umgang mit Informationen. Unser heutiges Verhaltensmuster in bezug auf die Information ist noch weit davon entfernt, deren Wert in einer künftigen Gesellschaft, bei der die Informationstechnik ebenso – wie das Verkehrswesen und die Energiewirtschaft schon jetzt – zur Infrastruktur zählt, zu entsprechen. Überlegungen zu ethisch-moralischen Normativen wurden nach dem INTERNET-Vorfall insbesondere in den USA angestellt. Äußerst fraglich sind daher solche Aktionen, die Preise für einen *unknackbaren* Virus ausschreiben /12/.

Viren und Recht

Viren verändern Daten und Programme. Mit diesem Tatbestand ist die strafrechtliche Relevanz für denjenigen, der Computerviren entwickelt und in Umlauf setzt, eindeutig gegeben. Das Strafgesetzbuch der BRD (Paragraphen 303a und 303b) sieht beispielsweise Freiheitsentzug von bis zu 5 Jahren vor.
Nach dem Zivilrecht kann derjenige, der Schaden vorsätzlich oder fahrlässig verursacht, dafür haftbar gemacht werden. Das trifft für Schäden, die durch Viren verursacht werden, natürlich genauso zu.
Es gibt – vielleicht bis auf die Ausnahme der Forschungen zu Fragen der sich selbst reproduzierenden Software – keinerlei Gründe, Viren zu entwickeln, geschweige denn, sie zu verbreiten. Jeder, der es dennoch tut, sollte sich auch über die straf- bzw. zivilrechtlichen Konsequenzen im klaren sein. Selbst der beste Vorsatz, äußerst sorgfältig mit entwickelten Viren umzugehen, sind – das hat die Praxis in bitterer Weise gelehrt – auf Sand gebaut.

Literatur

/1/ Blick in die Welt, Sächsische Zeitung vom 5./6. 11. 88
/2/ Bedrohungsinfekt, Horizont (1989) 4, S. 30
/3/ Computerviren. Mikroprozessortechnik, Berlin 2 (1989) 9, S. 259
/4/ Computer-Virus per Post, Sächsische Zeitung vom 15. 12. 1989
/5/ Erpresser-Software per Post – Eine erste Analyse, Datenschutzberater Düsseldorf, Frankfurt (1990) 1, S. 1
/6/ Cohen, F.: Computer Viruses – Theory and Experiments. Computer Security: a global challenge, North-Holland, Amsterdam, New York, Oxford, S. 143
/7/ Burger, R.: Das große Computer-Viren-Buch. DATA BECKER GmbH, Düsseldorf 1987
/8/ Schulz, R.: Informationen über Computerviren in der DDR. Vortrag auf einem Seminar zu Computerviren an der Martin-Luther-Universität Halle am 18. 1. 1990
/9/ Burger, R.: Alteration Searcher. DATA BECKER GmbH, Düsseldorf 1988
/10/ Software contra Computervirus, Mikroprozessortechnik, Berlin (1988) 7, S. 195
/11/ Wong, K.: Computer Virus – Nuisance or Corporate The rat? Datenschutz und Datensicherung (1989) 11, S. 567
/12/ Mußtopf, G.: Förderung der Entwicklung neuer Viren! Virus-Telex, Würzburg (1990) 4, S. 10

KONTAKT

Technische Universität Dresden, Informatikzentrum, Mommsenstr. 13, Dresden, 8027; Tel. 4 57 54 78

TERMINE

XIV. Mikroelektronik-Bauelementesymposium

WER? Kammer der Technik, Fachverband Elektrotechnik
WANN? Mai 1991
WO? Frankfurt(Oder)
WIE? Nähere Informationen bitte unter: Kammer der Technik, Clara-Zetkin-Straße 115–117, Berlin, 1086; Tel. 2 26 52 31 oder 2 26 53 85

Fachtagung Computer- und Mikroprozessortechnik

WER? Kammer der Technik, Fachverband Elektrotechnik
WANN? 1. Quartal 1991
WO? Berlin
WIE? Nähere Informationen bitte unter: Kammer der Technik, Clara-Zetkin-Straße 115–117, Berlin, 1086; Tel. 2 26 52 31 oder 2 26 53 85

Pollei

Grundlagen des logischen Programmierens Teil 1

Dr. Konrad Schultz, Chemnitz

Im Zuge der Entwicklung der Künstlichen Intelligenz verändert sich die Aufgabe des Computers: Statt vordergründig ein System von Handlungsvorschriften (einen Algorithmus) auszuführen, das der Programmierer entworfen hat, wird dem Computer eine genaue Beschreibung der Aufgabenstellung (die Spezifikation des Problems) samt der zu treffenden Voraussetzungen übergeben. Die Lösung des Problems hat der Computer dann selbständig zu finden.
Während bisher die Spezifikation des Problems nur eine vorbereitende Tätigkeit des Programmierers war – eine Voraussetzung für das weitere – und die Entwicklung eines geeigneten Algorithmus der Hauptinhalt war (die Codierung des Algorithmus ist schon früher vom Computer übernommen worden), so wird die Spezifikation nun Hauptinhalt des Programmierens. Der Programmierer hat das gestellte Problem soweit zu analysieren, daß es durch eine überschaubare Menge von Formeln mit genau definierbarer Bedeutung beschrieben wird. Diese Formeln sind solche der mathematischen Logik. (Eine weitere Entwicklung geht dahin, diese Forderungen abzuschwächen und dem Computer Formulierungen des Problems nur noch in natürlicher Sprache anzubieten.) Der Computer hat dann die Formeln zu verarbeiten, zu ermitteln, welche Objekte eine Lösung des Problems ergeben. Das geschieht wiederum mit Hilfsmitteln der mathematischen Logik, die somit auch die Algorithmen zur Bestimmung der gesuchten Objekte zur Verfügung stellt. Daher der Begriff *logische Programmierung*.
Wenn der Programmierer von der Entwicklung eines problemangepaßten Algorithmus entlastet wird und statt dessen der Computer (der natürlich nach einem bestimmten Algorithmus arbeitet) auf allgemeingültige Algorithmen der mathematischen Logik zurückgreift, wird für das einzelne Problem eine weniger effiziente Lösung entstehen. Das wird im Laufe der Zeit durch bessere Hardware-Leistungen weniger fühlbar und im übrigen dadurch ausgeglichen, daß größere Klassen von Problemen einheitlich und damit effektiver gelöst werden können.
Zum Verständnis der logischen Programmierung gehört also vor allem die Kenntnis einiger Grundbegriffe und Methoden der mathematischen Logik. Wir werden diese zusammenstellen und für die programmiersprachliche Umsetzung vorzugsweise auf Prolog zurückgreifen. Prolog – das **Pro**gramming in **Log**ic abgekürzt – wurde Anfang der siebziger Jahre von A. Colmerauer in Marseille für die Verarbeitung natürlicher Sprachen entwickelt und ist heute die am weitesten verbreitete Sprache der Logikprogrammierung.

Einfache Aussagen

Aussagen sind sprachliche Gebilde, von denen man sagen kann, daß sie wahr oder falsch sind:

7 ist eine Primzahl
Max und Moritz sind Freunde.
Hans liebt Grete.
Es ist kalt.

Die erste Aussage ist offenbar wahr, bei den beiden nächsten müßten wir uns zunächst verständigen, wer Max, Moritz, Hans, Grete sind, bei der letzten über Zeit und Ort. Ob eine Aussage wahr oder falsch ist, hängt also davon ab, wie wir sie inhaltlich verstehen. Daß wir die erste Aussage ohne weiteres als wahr ansehen, liegt auch nur daran, daß wir von unserer Schulbildung her eine einheitliche und genaue Auffassung der Begriffe *7* und *Primzahl* haben. Wir sehen es als unwesentlich an, ob die Entscheidung *wahr* oder *falsch* leicht fällt oder problematisch ist, ob das zwischen Hans und Grete schon wirkliche Liebe ist.
Grammatisch zerfällt jede einfache Aussage in Gegenstände (Individuen; Subjekte oder Objekte, zwischen denen nur ein formaler, kein logischer Unterschied besteht) und die Behauptung (das Prädikat) über diese Gegenstände. (Das ist auch für zusammengesetzte Aussagen richtig, dort interessieren uns aber zunächst die Zerlegungen in Teilaussagen oder Aussagenformen.)

Das Prädikat *ist eine Primzahl* trifft zu auf das Individuum *7*.
Das Prädikat *sind Freunde* trifft zu auf die *Gegenstände* (Subjekte) *Max* und *Moritz*.
Das Prädikat *liebt* trifft zu auf die Subjekte *Hans* und *Grete*.
Das Prädikat *kalt sein* trifft zu schlechthin.

Wir befreien die Aussagen von allem sprachlichen Ballast, indem wir einfach notieren

```
primzahl(7).
freunde(max,moritz).
liebt(hans,grete).
kalt.
```

In dieser Form werden die einfachen Aussagen auch in Prolog notiert und als Fakten in den Computer eingegeben. Prologspezifisch ist, daß die konstanten Namen *primzahl*, ..., *grete*, *kalt* mit Kleinbuchstaben beginnen. Ein solcher Fakt wird nach der Eingabe vom Computer als wahr angesehen, selbst dann, wenn er inhaltlich falsch ist; die Eingabe

```
primzahl(9)
```

wird nicht zurückgewiesen. Die Eingabe falscher Aussagen ist in Prolog gar nicht vorgesehen. Man kann in Prolog dem Computer nicht vorschreiben, daß er diese oder jene Aussage als falsch ansehen soll. Wenn eine Aussage nicht aus der Wissensbasis des Computers folgt und in diesem Sinne als nicht-wahr anzusehen ist, ist das zum einen etwas anderes, und zum anderen gibt es keinen Computeralgorithmus, um allgemein eine solche Nicht-Wahrheit nachzuweisen.
Wesentlich für den Computer ist nur der strukturelle Aufbau der Fakten, die inhaltliche Bedeutung kann er nicht begreifen. In unseren Beispielen bestehen sie aus

1. einem konstanten Namen, der als Prädikatenkonstante angesehen wird
2. in der Regel einer Argumentliste von konstanten Namen, die als Individuenkonstanten angesehen werden
3. einem Punkt.

Der Punkt hat dabei nur die Aufgabe, verschiedene Fakten (oder allgemeiner Klauseln) zu trennen. Der Fakt ohne den Punkt ist ein Atom. Ein Atom ist ein konstanter Name, dem eventuell eine Argumentliste folgt. (Fakten ohne Argumentliste sind die Ausnahme.) Wenn eine Argumentliste folgt, ist das Atom eine Struktur. Die Argumentliste wird von runden Klammern eingegrenzt, und die Listenelemente (Argumente) werden durch Kommas getrennt. Die Argumente können beliebige Atome sein, insbesondere auch selbst Strukturen sein. Das hängt mit dem Termbegriff zusammen, mit dem wir uns nun beschäftigen wollen.

Grundterme

Eine Aussage ist auch
Hans liebt seine Mutter.
beziehungsweise
Das Prädikat *liebt* trifft zu auf die Individuen *Hans* und *Mutter_von(Hans)*.
oder
liebt(hans,mutter_von(hans)).

mutter_von(hans) ist dabei eine Struktur, hat also die Gestalt einer Aussage (ohne Punkt freilich), wird aber von uns nicht als wahr oder falsch angesehen, ebensowenig vom Computer. *mutter_von(hans)* bezeichnet vielmehr ein Individuum, das durch das Individuum *hans* eindeutig bestimmt ist.
mutter_von(Mensch) bezeichnet die Mutter eines beliebigen, gegebenen Menschen *Mensch*, ebenso mag *produkt(Zahl1,Zahl2)* das eindeutig bestimmte Produkt zweier Zahlen, Zahl1 und Zahl2, bezeichnen.
Grundterme (oder konstante Terme) sind sprachliche Gebilde, die bestimmte Individuen bezeichnen. In formalisierter Form sind sie Atome, entweder konstante Namen, gedeutet als Individuenkonstanten, Zahlen (in Prolog von sekundärer Bedeutung) oder Strukturen aus einem konstanten Namen, gedeutet als Funktionskonstanten, und aus einer Argumentliste von Atomen, gedeutet als Grundterme. In formalisierter Form sind dabei Fakten (ohne Punkt) und Grundterme dasselbe. Der konstante Name am Anfang wird bei Fakten als Prädikatenkonstante, bei Grundtermen als Individuenkonstante bzw. Funktionskonstante gedeutet. Die Argumente werden dagegen in beiden Fällen als Grundterme gedeutet. Somit sind Grundterme Bestandteile von Fakten, aber nicht umgekehrt.

Generalisierungen und Variablen

Betrachten wir die Aussage „Jeder Mensch ist sterblich." Diese ist gleichwertig mit: „Für jeden Menschen X ist sterblich(X) wahr". In der mathematischen Logik schreibt man dafür

$\forall X$ sterblich(X)

Solch eine zusammengesetzte Aussage entsteht so: Wir nehmen eine Aussage her, hier etwa die einfache Aussage *sterblich (sokrates)*, ersetzen konstante Namen, hier *sokrates*, an einigen Stellen der Argumentliste durch variable Namen (kurz: Variablen), hier *X*, und es entsteht zunächst eine Aussagenform, hier *sterblich(X)*.
Eine Aussagenform ist ein sprachliches Gebilde mit Variablen, aus dem eine Aussage entsteht, indem für die Variablen konstante Namen für Gegenstände eines bestimmten Bereichs oder auch Zahlen eingesetzt werden. So ist in unserem Beispiel (aber nicht notwendig) sterblich(X) eine Aussagenform für den Bereich der Menschen, ebenso ist liebt(X,Y) eine Aussagenform für den Bereich

der Menschen, und aus diesen Aussagenformen entstehen wahre oder falsche Aussagen, wenn an die Stelle der Variablen Namen von Menschen treten: sterblich(shakespeare), liebt(romeo,julia), liebt(sokrates, xanthippe), die letzte Aussage soll falsch sein. Die Angabe eines Gegenstandsbereichs (Individuenbereichs) für die Variablen ist wesentlich: In der Aussagenform „X ist Einwohner von Y", notiert als *einwohner(X,Y)*, bezieht sich die Variable X vernünftigerweise auf Menschen, die Variable Y dagegen auf Ortschaften, *einwohner(goethe ,frankfurt)* war wahr, *einwohner(frankfurt, goethe)* (Frankfurt ist Einwohner von Goethe) dagegen sinnlos.

Aus einer Aussagenform entsteht aber auch eine (zusammengesetzte) Aussage, indem alle vorkommenden Variablen nach dem Beispiel von

∀X∀Y bruder(X,Y)

mit Hilfe von ∀ generalisiert werden. Um die entstandene Aussage sinnvoll lesen zu können, muß man wieder wissen, auf welche Individuenbereiche sich die Variablen beziehen, im Beispiel beziehen sich beide Variablen natürlicherweise auf den der Menschen, und die Aussage ∀X∀Y bruder(X,Y) ist dann zu lesen als

Für jeden Menschen X und jeden Menschen Y ist bruder(X,Y) wahr.

also

Alle Menschen sind Brüder.

Die Angabe eines Individuenbereichs ist für solche generalisierten Aussagen noch wesentlicher als für Aussagenformen:

∀X positiv(produkt(X,X))

(Für jedes X gilt: Das Produkt von X mit sich selbst ist positiv) ist wahr, wenn als Individuenbereich für die Variable X der Bereich der reellen Zahlen gewählt wird, aber falsch, wenn der Bereich aller komplexen Zahlen zum Individuenbereich wird.

Ein variabler Name oder eine Variable beginnt in Prolog mit einem Großbuchstaben, wir halten es in dieser Übersicht ebenso. Die Generalisierung, in der mathematischen Logik ausgedrückt durch ∀, gefolgt von den Variablen, wird in Prolog nicht dargestellt. Jede vorkommende Variable (außer in Fragen) wird als generalisiert angesehen. So wird durch die Eingabe von

sterblich(X).

der Computer angewiesen, die Aussage „Jedes X ist sterblich" bzw. „Alle X sind sterblich" als wahr anzusehen. Dies bedeutet: Der Computer sieht alle (Grund-)Beispiele von sterblich(X) bzw. von ∀X sterblich(X) als wahr an, also alle Aussagen, die aus der Aussagenform sterblich(X) durch die Einsetzung von (Grund-) Termen für die Variable X entstehen. Falls ihm der Fakt

liebt(hans,mutter_von(eva)).

eingegeben wurde, nimmt er dann auch sterblich(hans), sterblich(eva), sterblich(mutter_von(hans)) usw. als wahr an, so als wären auch die Fakten

sterblich(hans).
sterblich(eva).

und so weiter eingegeben worden. Für den Computer ist dabei für sämtliche Variablen der Individuenbereich grundsätzlich der aller Grundterme, schließlich ist für ihn der Bereich der Namen (Zeichenketten) der einzige, den er unmittelbar überblicken kann: Er führt über alle vorgekommenen konstanten Namen eine Liste, mit deren Hilfe dann der im allgemeinen unendliche Bereich der Grundterme generiert werden kann.

Ich erwähnte schon, daß die Betrachtung von endlich vielen Grundbeispielen theoretisch grundsätzlich ausreicht. Dies ermöglicht die Lösung von logisch formulierten Problemen durch den Computer.

Es ist völlig gleichwertig, ob wir eingeben

sterblich(X).

oder

sterblich(Mensch).

denn auf die Bezeichnung der Variablen kommt es nicht an. Es ist üblich, durch die entsprechende Wahl der Bezeichnungen Programme zu kommentieren.

Variablen sind in Prolog nur lokal (d.h. innerhalb einer Klausel) gültig. In verschiedenen Klauseln kennzeichnet ein und derselbe variable Name inhaltlich unterschiedliche Variablen. So können die beiden Fakten

sterblich(Mensch).
irrt(Mensch).

ebensogut

sterblich(X).
irrt(Y).

geschrieben werden, drücken sie doch die Aussagen

Jeder Mensch ist sterblich, und jeder Mensch irrt.

bzw. gleichwertig

Jedes X aus dem Bereich der Menschen ist sterblich, und jedes Y aus dem Bereich der Menschen irrt.

aus. In Prolog ist das Unterstreichungszeichen _ allein (das als Großbuchstabe gerechnet wird) die sogenannte anonyme Variable. Sie wird gern verwendet, weil intern für sie kein Speicherplatz belegt wird. Und zwar können alle in einer Klausel nur je einmal auftretenden Variablen durch die anonyme ersetzt werden.

sterblich(_).

ist gleichwertig mit

sterblich(Mensch).

und

bruder(_,_).

ist gleichwertig mit

bruder(Mensch,Mitmensch).

(mit zwei Variablen!).

Partikularisierungen und allgemeine Terme

Betrachten wir einmal die Aussage „Max hat einen Freund." Diese ist gleichwertig mit „Es gibt einen Menschen X derart, daß freund(max,X) wahr ist." In der mathematischen Logik schreibt man dafür

∃X freund(max,X)

Auch hier bezieht sich die Variable X auf einen bestimmten Individuenbereich, vernünftigerweise den der Menschen. Während aber die Aussage

∀X freund(max,X)

wahr ist, wenn Max mit jedem Individuum aus diesem Bereich befreundet ist, ist

∃X freund(max,X)

wahr, sofern es wenigstens ein Individuum aus diesem Bereich, also einen Menschen, gibt, mit dem Max befreundet ist. Während ∀ die Generalisierung (die **All**behauptung) bezeichnet, bezeichnet ∃ die **Partikularisierung** die (**Existenz**behauptung). Der Oberbegriff für Generalisierung und Partikularisierung ist **Quantifizierung.**

Eine partikularisierte Aussage kann in PROLOG nicht formuliert werden. PROLOG verlangt konstruktive Eingaben (und wird dafür auch konstruktive Antworten liefern). Und

∃X freund(max,X)

ist nur dann konstruktiv wahr, wenn wir ein Individuum für die Variable X benennen können, welches die Aussagenform freund (max,X) zu einer wahren Aussage macht. Wir können also den Computer in PROLOG nur dadurch mit der Wahrheit von

∃X freund(max,X)

bekannt machen, daß wir einen Fakt wie

freund(max,moritz).

eingeben.

„Jeder Mensch hat einen Freund." – Diese Aussage logisch korrekt zu notieren, macht jetzt keine Schwierigkeiten

∀X∃Y freund(X,Y)

Dem Computer kann man sie aber nur beibringen, indem man solche Y angibt, abhängig jetzt freilich von den X, etwa in der Form

freund(X,vater_von(X)).

mit der Bedeutung „Jeder Mensch ist mit seinem Vater befreundet." „vater_von" ist hier ein konstanter Name, der als Funktionenkonstante zu deuten ist. So wie nun vater_von(max) ein Term ist, ist auch vater_von(X) ein Term, freilich kein konstanter Term, kein Grundterm, sondern einer mit Variablen statt einiger Konstanten, ein variabler Term. Variable Terme werden zu Grundtermen, indem für die vorkommenden Variablen Namen von Gegenständen (Individuenkonstanten) eingesetzt werden, und bezeichnen dann selbst Gegenstände. Variable Terme verhalten sich zu Grundtermen insoweit also analog wie Aussagenformen zu Aussagen.

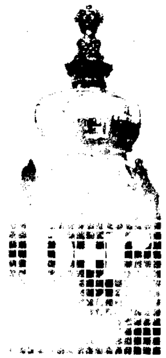
wird fortgesetzt

✉ KONTAKT ☎

Technische Universität Chemnitz, Sektion Informatik, Postfach 964, Chemnitz, 9010; Tel. 66 82 52

„In sieben Sekunden müßte wieder einer anbeißen!"

Zeichnung: Steger



Hardware
Software
Kopierer
Papier

Telefon 487 4103/4308
Telex 2 63 98

b + r CompuSoft Systeme GmbH
Leningrader Str. 9
Dresden
8010

aus unserem Angebot

80286, IBM-AT kompatibel **Preis: 3.999,00 DM**

80386 SX 16 MHz, 2 MB (erweiterbar auf 8 MB)
40 MB Festplatte, 3,5" Laufwerk 1,44 MB, VGA Color, DOS 3.3
Preis: 5.747,00 DM

80386 25 MHz, 4 MB (erweiterbar auf 16 MB), 32 KB Cash,
40 MB Festplatte, 3,5" Laufwerk 1,44 MB, VGA Color, DOS 3.3
Preis: 8.915,00 DM

Laptop 80286, 12,5 MHz, 640 MB(erweiterbar auf 2,6 MB)
40 MB Festplatte, 3,5" 1,44 MB, EGA-Plasmamonitor, DOS 3.3
Preis: 5.199,00 DM

Festplatten:
Seagate 251-1 42,8 MB, 28 ms **Preis: 746,00 DM**
Seagate 157R-1 49,1 MB, 28 ms **Preis: 789,00 DM**
Seagate 4096-2 83,9 MB, 28 ms **Preis: 1.326,00 DM**

Drucker:
Epson LQ 400, 24 Nadeln, Centronics **Preis: 876,00 DM**
Canon Laserdrucker A4, 1,5 MB, 4 S/min **Preis: 4.356,00 DM**

Streamer:
Archive ST 600, intern Contr./Kass.,60 MB **Preis: 1.544,00 DM**
Tandberg TDC 3820, intern SCSI, 320 MB **Preis: 2.208,00 DM**

SW - Angebot des Monats

Zur Lösung betriebswirtschaftlicher Aufgaben in mittelständischen Unternehmen ist das integrierte Softwarepaket

DER PC - KAUFMANN

sehr gut geeignet, denn es vereinigt die Lösung aller anfallenden Aufgaben in sich:

- **Finanzbuchhaltung**
- **Tabellenkalkulation**
- **Bestellwesen**
- **Geschäftsgrafik**
- **Auftragsabwicklung**
- **Textverarbeitung**
- **Lohnrechnung**

Wegen der übersichtlichen Dialogführung durch den Rechner, mittels der auf dem Bildschirm anzeigbaren Hilfstexte und durch das Vorhandensein einer leicht gefaßten und anschaulichen Dokumentation ist eine einfache Einarbeitung und Handhabung der Software gegeben.

Nutzungsvoraussetzungen:
IBM-PC oder 100% Kompatibler, 640 KB Arbeitsspeicher, Diskettenlaufwerk 5,25 oder 3,5 Zoll, Festplattenspeicher 20 MB, MS-DOS ab Version 3.3.

Der PC-Kaufmann
Preis: 899,00 DM

PC - Angebot des Monats

Goldstar 80286, 16/8 MHz, 1 MB RAM, 40 MB Festplatte
5,25" Floppylaufwerk, VGA Monochrom, MS-DOS 3.3

Preis: **3.999,00 DM**

mit VGA Color Schirm

Aufpreis: **500,00 DM**

Lisp

Teil 2

Dr. Ulrich Kriegel

Rekursionen

Programmierung basiert auf der Auswahl von Kontrollstrukturen, die festlegen, wie die einzelnen Programmschritte ausgeführt werden sollen. Ein typisches Beispiel für eine derartige Kontrollstruktur in Lisp ist die Rekursion. Der Grundgedanke dabei ist, daß eine Funktion ein Problem etwas vereinfacht und dieses dann an eine oder mehrere identische Kopien von sich selbst zur weiteren Verarbeitung übergibt. In Abhängigkeit davon, ob bei einem Rekursionsschritt die Funktion ein- oder mehrmals aufgerufen wird, spricht man von Einfach- bzw. Mehrfach-Rekursionen.

```
(defun fibonacci
       (n)
   (cond ;; Abbruch n <= 1
         ((>= 1 n) 1)
         ;; doppelte Rekursion
         (t (+ (fibonacci (- n 1))
               (fibonacci (- n 2))))))
```

Bild 3 Definition der Funktion Fibonacci

```
-> (FIBONACCI 4)
I-> (FIBONACCI 3)
II-> (FIBONACCI 2)
III-> (FIBONACCI 1)
III<- (FIBONACCI 1)
III-> (FIBONACCI 0)
III<- (FIBONACCI 1)
II<- (FIBONACCI 2)
II-> (FIBONACCI 1)
II<- (FIBONACCI 1)
I<- (FIBONACCI 3)
I-> (FIBONACCI 2)
II-> (FIBONACCI 1)
II<- (FIBONACCI 1)
II-> (FIBONACCI 0)
II<- (FIBONACCI 1)
I<- (FIBONACCI 2)
<- (FIBONACCI 5)
```

Bild 4 Rease-Protokoll des Aufrufs (Fibonacci 4)

```
(defun mp-copy-list
       (li)
 (cond ((null li) nil)
       (t (cons (car li)
          (mp-copy-list (cdr li)))))).
```

Bild 5 Definition der Funktion MP-copy-list

```
-> MP-COPY-LIST ((A B C))
I-> MP-COPY-LIST ((B C))
II-> MP-COPY-LIST ((C))
III-> MP-COPY-LIST (NIL)
III<- MP-COPY-LIST (NIL)
II<- MP-COPY-LIST ((C))
I<- MP-COPY-LIST ((B C))
<- MP-COPY-LIST ((A B C))
```

Bild 6 Trace-Protokoll des Aufrufs (mp-copy-list '(A B C))

```
(defun mp-member
       (item li)
  (cond ;;Liste leer ?
        ((null li) nil)
        ;;ITEM gefunden ?
        ((equal item  (car li)) li)
        ;;Rekursion
        (t(mp-member item (cdr li)))))
```

Bild 7 Definition der Funktion MP-member

Eine rekursive Funktion muß immer die folgenden induktiven Aspekte des zu behandelnden Problems enthalten:

① Es muß ein Test auf eine Abbruchbedingung existieren, bei der die Rekursion abgebrochen wird.

② Es muß eine Methode für die Vereinfachung des Problems existieren, so daß die Folge der vereinfachten Probleme gegen die Abbruchbedingung konvergiert.

Im folgenden Abschnitt sollen die möglichen Formen der Rekursion und ihre Anwendung bei der Synthese und Analyse von S-Ausdrücken behandelt werden.

Rekursive Synthese von S-Ausdrücken

Als Beispiel für eine einfach-rekursive Funktion soll die in Bild 1 gezeigte Funktion MUL dienen, die zwei positive ganze Zahlen A und B multipliziert. Die Funktion bricht ab (terminiert), da B schrittweise um Eins verkleinert und damit einmal der Test (= B O) erfüllt wird. Die Funktion ist korrekt, denn es gilt:

$A * B = A + (B - 1) * A$

Das in Bild 2 gegebene Trace-Protokoll zeigt die Abarbeitung von (MUL 4 2). Da auf jedem Rekursionsniveau die Funktion MUL nur einmal aufgerufen wird, ist der Aufrufbaum linear. Aus den zurückgegebenen Funktionswerten ist ersichtlich, daß das Gesamtergebnis aus den Teilergebnissen zusammengesetzt wird. Man spricht deshalb auch von rekursiver Synthese.

Ein Beispiel für eine doppelt-rekursive Funktion soll anhand der Berechnung der sogenannten Fibonacci-Zahlen gegeben werden. Leonardo von Pisa, auch Fibonacci genannt, stellte ein diskretes Modell für das Wachstum einer Population von Hasen auf, das durch die folgende Gleichung beschrieben wird:

$f(n) = f(n - 1) + f(n - 2)$

wenn $n > 1$ ist, sonst gilt $f(n) = 1$.

Bild 3 zeigt die Funktion Fibonacci, Bild 4 das Trace-Protokoll des Aufrufs von (Fibonacci 4). Der Aufrufbaum ist nicht mehr linear wie im vorhergehenden Beispiel, da auf jeder Rekursionsebene zwei Kopien der Funktion Fibonacci aufgerufen werden.

Nach der Lösung dieses relativ einfachen numerischen Problems wollen wir uns nun wieder der Symbolverarbeitung zuwenden. Ein weiteres Beispiel für die rekursive Synthese von S-Ausdrücken ist eine Funktion, die Listen auf dem obersten Niveau kopiert. Da die in Common Lisp definierte Funktion Copy-List etwas allgemeiner anwendbar ist als die folgende Definition, wollen wir sie MP-copy-lisp nennen (Bild 5). Allgemein wird in diesem Kurs immer dann ein Funktionsname mit dem Präfix MP versehen, wenn eine Funktion gleichen Namens im Common-Lisp-Standard existiert, aber die hier vorgestellte Funktion im Vergleich zum Standard nur eingeschränkt verwendbar ist. Die Auswertung von (MP-copy-list '(A B C)) liefert das in Bild 6 dargestellte Aufrufprotokoll.

```
(defun mul
       (a b)
  (cond ;; b = 0 ?
        ((= b 0)0)
        ;; Rekursion
        (t (+ a (mul a (- b 1)))))
```

Bild 1 Definition der Funktion Mul

```
-> MUL (4 2)
I-> MUL (4 1)
II-> MUL (4 0)
II<- MUL (0)
I<- MUL (4)
<- MUL (8)
```

Bild 2 Trace-Protokoll des Aufrufs (Mul 4 2)

Aufgabe 4: Schreiben Sie eine Funktion, die rekursiv die Länge einer Liste bestimmt. (Die Lösungen der Aufgaben dieses Teils finden Sie ausnahmsweise erst im Teil 3 in MP 11/90.)

Restrekursion – die rekursive Analyse von S-Ausdrücken

Ebenso wie zur Synthese können Rekursionen auch zur Analyse von S-Ausdrücken benutzt werden.

Nehmen wir einmal an, daß in unserem Lisp die Grundfunktion Member des Lisp-Standards nicht vorhanden ist. Member testet, ob das erste Argument auf dem obersten Niveau in der Liste enthalten ist, die als zweites Argument übergeben wird (Bild 7). MP-Member behandelt zwei Basissituationen: Entweder ist die Liste leer, dann wird NIL zurückgegeben, oder ITEM ist das erste Element der Liste LI, dann wird LI zurückgegeben. Tritt keine dieser Basissituationen auf, so übergibt MP-Member ein vereinfachtes Problem, nämlich die Suche von ITEM in der um ein Element verkürzten Liste, an eine Kopie von sich selbst. In Bild 8 ist die Abarbeitung des Aufrufs von (MP-Member 'C '(A B C D E F)) dargestellt. Beachten Sie bitte, daß MP-Member entweder eine Basissituation erkennt oder das Problem *delegiert*. Im Gegensatz zu MP-copy-list müssen deshalb auch keine Zwischenergebnisse gespeichert werden, während die Kopie arbeitet. Der Wert, den MP-Member zurückgibt, ist genau der Wert, den die Kopie auf dem untersten Niveau der Rekursion liefert.

```
-> MP-MEMBER (C (A B C D E F))
I-> MP-MEMBER (C (B C D E F))
II-> MP-MEMBER (C (C D E F))
II<- MP-MEMBER ((C D E F))
I<- MP-MEMBER ((C D E F))
<- MP-MEMBER ((C D E F))
```

Bild 8 Trace-Protokoll des Aufrufs (MP-member 'C '(A B C D E F))

Allgemein nennt man eine Funktion *restrekursiv*, wenn der zurückgegebene Wert entweder direkt berechnet wurde oder (wie oben) der Wert des rekursiven Aufrufs auf dem untersten Rekursionsniveau ist. Restrekursive Funktionen müssen den Wert, den die rekursiven Aufrufe liefern, nicht weiter bearbeiten. Vom Prinzip her muß deshalb die Kopie auch nicht zu der sie aufrufenden Kopie (bzw. zum Original) zurückkehren; statt dessen kann direkt zu der rufenden Funktion verzweigt werden. Optimierende Compiler, z. B. der XCL-Compiler, nutzen diese Tatsache aus und erzeugen für restrekursive Funktionen äußerst effektiven Maschinenkode. Das ist auch der Grund, warum von vielen Programmierern der Programmierstil mit restrekursiven Funktionen bevorzugt wird. Mit einem Trick ist es sogar möglich, eine bestimmte Klasse nicht restrekursiver Funktionen in restrekursive Funktionen umzuwandeln.

Restrekursion mit Akkumulatorvariable

Bild 9 zeigt die Definition der Funktion Copy-N, die aus einer Liste die Teilliste der maximal ersten n Elemente erzeugt. Die Funktion ist nicht restrekursiv, weil sie das Ergebnis des Aufrufs der Kopie der Funktion Cons übergibt, um die Ergebnisliste zu bilden.

Rufen wir, wie in Bild 10 dargestellt, statt Copy-n eine Hilfsfunktion rekursiv auf, so kann diese restrekursiv programmiert werden. Der Trick besteht darin, daß die Teilergebnisse des jeweiligen Rekursionsniveaus dem nächsten Niveau mit übergeben werden. Die Hilfsfunktion ist restrekursiv, weil der Wert des rekursiven Aufrufs in Cond direkt als Rückgabewert benutzt wird.

Generell ist eine Restrekursion mit Akkumulatorvariable immer dann möglich, wenn bei jeder Rekursion ein Teilergebnis bestimmbar ist. Diese Tatsache nutzen optimierende Compiler aus. Bei der Kode-Optimierung werden dann automatisch restrekursive Hilfsfunktionen generiert.

Aufgabe 5: Definieren Sie die Funktion MP-reverse, die die oberste Ebene einer Liste umdreht, als restrekursive Funktion mit Akkumulatorvarable.

Rückwärtsrekursion – die Synthese einer Liste vom Ende her

Zur Lösung mancher Probleme ist es oft günstig, eine Liste vom Ende her abzubauen. Die naheliegenste Lösung hierbei wäre wohl, die Liste erst mit Reverse umzudrehen und dann zu bearbeiten. Bei langen Listen ist dieses Verfahren jedoch zeit- und speicherplatzaufwendig. Einen Ausweg bietet hier die Rückwärtsrekursion, die im folgenden am Beispiel der Funktion Copy-last erläutert werden soll (Bild 11). Copy-last kopiert die letzten n Elemente einer Liste.

Zum Verständnis der Arbeitsweise rückwärtsrekursiver Funktionen ist es jedoch notwendig, sich etwas eingehender mit dem Gültigkeitsbereich von Variablen in Common Lisp zu beschäftigen. Normalerweise hat in Common Lisp eine Variable, die in der Parameterliste einer Funktion auftritt oder in der Funktion als Hilfsvariable spezifiziert wird, einen lexikalischen Gültigkeitsbereich; das bedeutet, daß sie innerhalb der Funktion frei verfügbar ist, ihre Bindung aber außerhalb der Funktion unbestimmt ist. Den Gegensatz dazu bilden dynamisch gebundene Variablen (auch Special-Variablen genannt), die einen dynamischen Gültigkeitsbereich besitzen, das heißt, sie existieren so lange, wie das sie etablierende Konstrukt existiert. Wird nun die Variable **n** innerhalb der Funktion Copy-last durch **(declare (special n))** als dynamisch gebundene Variable deklariert, so können alle Funktionen, die von dieser Funktion aufgerufen werden – in diesem Falle alle Kopien der Hilfsfunktion Copy-last-aux, auf diese Variable zugreifen. Diese Tatsache wollen wir bei der Ablaufsteuerung ausnutzen: Die Hilfsfunktion soll die Liste rekursiv abbauen und dann vom Ende beginnend, das Ergebnis synthetisieren. Dazu werden die Ergebnisse der einzelnen Kopien der Hilfsfunktionen in einer lokalen Variablen L1 abgelegt. Die mit dem letzten Element der Liste aufgerufene Kopie der Hilfsfunktion verringert (dekrementiert) n um Eins und gibt das letzte Element zurück, das an die Hilfsvariable L1 der übergeordneten Kopie gebunden ist. Daraus und aus dem ersten Element der Input-Liste wird eine neue Teilliste erzeugt, und n wird dekrementiert. Der Prozeß bricht ab, wenn n gleich Null ist. (In unvollständigen Implementationen von Common Lisp, beispielsweise in den ersten Versionen von GC-Lisp, waren nur dynamische Variablenbindungen möglich. In diesem Fall muß die Deklaration als dynamische Variable entfallen.)

In der Definition Copy-last-aux ist die Spezialform Let zur Deklaration lokaler Variable enthalten, die die folgende Syntax besitzt.

(LET ((VAR-1 INIT-1) . . . (VAR-n INIT-n))
(ANWEISUNG-1)
. . .
(ANWEISUNG-n))

VAR-1 bis VAR-n sind im Normalfall lokale lexikal gebundene Variablen, die durch die optionalen Formen Init-1 bis Init-n initialisiert werden. Die Initialisierung erfolgt parallel, das heißt, zuerst werden alle Initialisierungsformen ausgewertet, und danach erfolgt die Initialisierung. Werden die Variablen ohne Initialisierungsformen angegeben, so müssen die Variablennamen nicht geklammert werden. Nach der Liste (. . .(VAR-i INIT-i). . .) bzw. (. . .VARi. . .) kann optional eine Deklaration folgen, in der beispielsweise einige der Variablen als dynamisch gebundene Variable spezifiziert werden. Danach folgt der Körper der Let-Form, das heißt, beliebig viele Anweisungen, die sequentiell abgearbeitet werden. Der Gültigkeitsbereich der lexikalen Var-i ist auf den Körper der Let-Form beschränkt. Das Ergebnis der letzten Anweisung im Körper ist gleichzeitig das Ergebnis der Let-Form. Neben der Spezialform Let existiert noch die Spezialform Let* mit analoger Syntax, die jedoch im Gegensatz zu Let die Variablen sequentiell initialisiert.

Zur Veranschaulichung der Arbeitsweise von Copy-last-aux enthält Bild 12 ein Trace-Protokoll mit den zusätzlich eingetragenen Werten von n und L1.

```
(defun copy-n
       (li n)
 (cond ;;fertig ?
       ((= n 0) nil)
       ;;Rekursion
       (t(cons (car li)
          (copy-n (cdr-li)(- n 1))))))
```

Bild 9 Definition der Funktion Copy-n

```
(defun copy-n
       (li n)
 (copy-n-aux li n nil))

(defun copy-n-aux
       (li n acc)
 (cond ;;fertig ? -> umdrehen der Liste
       ;;im Akkumulator
       ((= n 0)(reverse acc))
       ;Restrekursion
       (t (copy-n-aux (cdr li)
             (- n 1)
             (cons (car li) acc)))))
```

Bild 10 Definition der Funktion Copy-n mit restrekursiver Hilfsfunktion

```
(defun copy-last
       (li n)
 (declare (special n)) ;Deklariere n
 (copy-last-aux li))   ;Aufruf der Hilfsfkt.

(defun copy-last-aux
       (l)
 (cond ((null l) nil)
       (t (let ((l1 (copy-last-aux
                     (cdr l))))
              (cond ((= 0 n) l1)
                    (t (setq n (- n 1))
                       (cons (car l)
                          l1)))))))
```

Bild 11 Definition der Funktion Copy-last

```
-> COPY-LAST-AUX ((A B C D))   N = 2  L1 = ?
I-> COPY-LAST-AUX ((B C D))    N = 2  L1 = ?
II-> COPY-LAST-AUX ((C D))     N = 2  L1 = ?
III-> COPY-LAST-AUX ((D))      N = 2  L1 = ?
IIII-> COPY-LAST-AUX (NIL)     N = 2  L1 = ?
IIII<- COPY-LAST-AUX (NIL)     N = 2  L1 = ?
III<- COPY-LAST-AUX ((D))      N = 2  L1 = NIL
II<- COPY-LAST-AUX ((C D))     N = 1  L1 = (D)
I<- COPY-LAST-AUX ((C D))      N = 0  L1 = (C D)
<- COPY-LAST-AUX((C D))        N = 0  L1 = (C D)
```

Bild 12 Trace-Protokoll des Aufrufs (copy-last-aux) '(A B C D))

Aufgabe 6: Schreiben Sie die Funktion MP-reverse, die die Elemente einer Liste umdreht, unter Ausnutzung der Rückwärtsrekursion.

Rekursive Analyse und Synthese

In den vorangegangenen Abschnitten haben wir Funktionen zur rekursiven Analyse und zur rekursiven Synthese symbolischer Ausdrücke getrennt behandelt. In der Programmierpraxis tritt jedoch häufig der Fall auf, daß rekursive Synthese und Analyse gekoppelt sind. Als Beispiel soll die Funktion MP-count dienen, die die Anzahl der Atome in einem S-Ausdruck feststellt (Bild 13). Die Funktion behandelt 2 Basisfälle, nämlich die leere Liste und das Atom. Treffen beide Fälle nicht zu, so wird MP-count auf Car und Cdr der Liste getrennt angewendet. Die Ergebnisse der Aufrufe werden dann addiert. Diese Funktion ist doppelt-rekursiv. Zur Illustration enthält Bild 14 das Trace-Protokoll des Aufrufs **(mp-count '(* a (sin b)))**.

```
(defun mp-count
      (l)
  (cond ;;Liste leer ?
        ((null l) 0)
        ;;atom ?
        ((atom l) 1)
        ;;analysiere car und cdr
        ;;und addiere Ergebnisse
        (t (+ (mp-count (car l))
              (mp-count (cdr l))))))
```

Bild 13 Definition der Funktion MP-count

```
-> (MP-COUNT (* A (SIN B)))
I-> (MP-COUNT *)
I<- (MP-COUNT 1)
I-> (MP-COUNT (A (SIN B)))
II-> (MP-COUNT A)
II<- (MP-COUNT 1)
II-> (MP-COUNT ((SIN B)))
III-> (MP-COUNT (SIN B))
IIII-> (MP-COUNT SIN)
IIII<- (MP-COUNT 1)
IIII-> (MP-COUNT (B))
IIIII-> (MP-COUNT B)
IIIII<- (MP-COUNT 1)
IIIII-> (MP-COUNT NIL)
IIIII<- (MP-COUNT 0)
IIII<- (MP-COUNT 1)
III<- (MP-COUNT 2)
III-> (MP-COUNT NIL)
III<- (MP-COUNT 0)
II<- (MP-COUNT 2)
I<- (MP-COUNT 3)
<- (MP-COUNT 4)
```

Bild 14 Trace-Protokoll des Aufrufs (mp-count '(a (sin b)))*

Allgemein kann gesagt werden, daß einfach-rekursive Funktionen die natürliche Methode sind, um mit Sequenzen, beispielsweise mit den obersten Elementen von List zu arbeiten. Die Verwendung doppelt-rekursiver Funktionen bieten sich dann an, wenn es darum geht, verschachtelte Listen auch in der Tiefe zu untersuchen, so daß dabei alle Elemente erreicht werden. Der Berechnungsaufwand für einfach-rekursive Funktionen ist dabei proportional der Länge der zu bearbeitenden Liste, während er für doppelt-rekursive Funktionen proportional zur Anzahl der Elemente in der Liste ist.

Der Terminplaner

Wir wollen nun mit der Entwicklung von Funktionen beginnen, die es erlauben, einen einfachen Terminplan zu verwalten. Terminpläne für bestimmte Personen sollen aufgestellt, freie Termine ermittelt, Termine in Pläne verschiedener Nutzer eingetragen oder eine Terminliste soll ausgegeben werden. Das Programm ist so angelegt, daß Sie es nach Ihren Wünschen selbst erweitern können. So wäre es beispielsweise denkbar, daß die Kalenderstrukturen in einer externen Datei gesichert werden. Bei jedem Start des Systems könnten sie dann automatisch geladen werden, und in Abhängigkeit vom Tagesdatum wird die Liste der nächsten Termine ausgegeben.

Der Terminplan weist folgende logische Struktur auf:

- Er besteht aus einer Liste von Tagesplänen.
- Ein Tagesplan soll das Tagesdatum und eine Liste von Terminen enthalten.
- Ein Tagesdatum setzt sich aus dem Wochentag, dem Tag und dem Monat zusammen.
- Ein Termin besteht aus Beginn- und Endzeitpunkt und aus dem Thema.
- Eine Zeitangabe enthält Stunden- und Minutenangaben.

Wir haben es also mit Objekten zu tun, die eine unveränderliche Struktur aufweisen. Nur die Inhalte der einzelnen Datenfelder, die auch Slots genannt werden, können sich jeweils ändern. Wir können derartige Strukturen natürlich mittels Listen darstellen. Auf die einzelnen Slots könnte man dann über die Funktionen der Car/Cdr-Familie zugreifen.

Man kann sich leicht vorstellen, wie gut lesbar dann Programme wären, die mit verschachtelten Strukturen arbeiten. Eine Änderung der Datenstruktur, beispielsweise der Reihenfolge der Slots, hätte schwerwiegende Folgen. Im gesamten Programm müßten beispielsweise Aufrufe von Car durch solche von Cadr ersetzt werden.

Dabei sind Fehler möglich, denn bestimmte Aufrufe von Car könnten auch zur Steuerung der Rekursion dienen.

Ein Ausweg aus dieser Misere besteht in der Definition abstrakter Datentypen, bei denen die physische Repräsentation der Daten dem Nutzer verborgen bleibt. Der Zugriff auf die einzelnen Slots einer solchen abstrakten Datenstruktur erfolgt über sogenannte Selektorfunktionen.

Programme, die abstrakte Datentypen enthalten, sind bedeutend leichter zu verstehen, und vor allen Dingen lassen sie sich leichter warten.

Ist das Datum eine Liste aus Wochentag, Tag und Monat, so könnte eine Selektorfunktion für den Monatsslot wie folgt aussehen:

(defun datum-monat(datum)
(caddr datum))

Zeiteinbußen durch die zusätzlichen Funktionsaufrufe bei der Verwendung abstrakter Datentypen können vermieden werden, wenn die entsprechenden Funktionen als Inline-Funktionen deklariert werden. Der Compiler kann diese Funktionsaufrufe dann durch den entsprechenden Inline-Kode ersetzen, beispielsweise (datum-monat datum) durch (caddr datum). Die Anzahl der Selektorfunktionen wächst linear mit der Zahl der Slots in der Struktur. Viele Dialekte von Lisp stellen deshalb dem Programmierer mit Defstruct ein Hilfsmittel zur Verfügung, das automatisch die entsprechenden Selektorfunktionen, ein Typprädikat, eine Konstruktorfunktion und eine Kopierfunktion erzeugt.

In Common Lisp hat der Aufruf von Defstruct im einfachsten Falle die folgende Syntax:

```
(defstruct name
      (slot-1 init-1)
      (slot-2 init-2)
      ...
      (slot-n init-n)).
```

Mit diesem Aufruf werden
- die Konstruktorfunktion *Make-name*,
- n Selektorfunktionen *Name-slot-i*,
- eine Kopierfunktion *Copy-name* und
- ein Typprädikat *Name-p*

generiert.

Der Aufruf von **(Make-name) erzeugt dann** eine Struktur mit n Slots, die mit den Werten Init-i initialisiert wurden. Werden keine Init-Werte angegeben, so können die Klammern um die Slotnamen entfallen. Die Slots werden dann mit NIL initialisiert.

Soll bei der Konstruktion der Struktur m Slots explizit ein Wert zugewiesen werden, so geschieht das durch den Aufruf (make-name :slot-i Wert-i). Symbole, die mit einem Doppelpunkt beginnen, werden auch Schlüsselwörter genannt. Diese haben sich selbst zum Wert und müssen deshalb nicht *quotiert* werden. Durch die Angabe der Schlüsselwörter zur Kennzeichnung der entsprechenden Slots ist die Reihenfolge der Schlüsselwort – Wert – Paare beliebig. Die physische Umsetzung der Struktur hängt von der Implementation ab. Nun stellt sich die Frage, wie man die Inhalte der einzelnen Slots manipulieren kann. Müßte es nicht auch dafür n Funktionen geben, für jeden Slot eine?

Um dieses Problem allgemein lösen zu können, wurde in Common Lisp das Konzept der verallgemeinerten Variablen eingeführt. Unter einer verallgemeinerten Variable wird dabei ein beliebiges Lisp-Objekt verstanden, das Daten aufnehmen kann und für das durch eine entsprechende Selektorfunktion ein Zugriffspfad definiert ist. So ist beispielsweise Cadr einer Liste Ll eine verallgemeinerte Variable, der Zugriffspfad ist (cadr li). Mit Hilfe von Setf kann eine verallgemeinerte Variable belegt werden:

(Setf <zugriffspfad><wert>).

Beispiel:
(setq li '(a b)) ==> (A B)
(setf (cadr li) 'x) ==> X
li ==> (A X).

Da die einzelnen Slots einer Struktur S verallgemeinerte Variablen sind, können deren Inhalte mit (Setf (<selektorfunktion> s) <wert>) geändert werden.

An dieser Stelle muß unbedingt darauf hingewiesen werden, daß Defstruct ein viel mächtigeres Werkzeug ist, als durch diesen Kurs gezeigt werden kann. So ist es möglich,

```
;;; Definition einer Zeitstruktur aus
;;; Stunde (h) und Minute (m)
(defstruct time (h 0) (m 0))

;;; Definition einer Datumsstruktur aus
;;; Wochentag (wd) Tag (day) und Monat (mon)
(defstruct date (wd 'mo) (day 1) (mon 'jan))

;;; Definition eines Termins aus Beginn
;;; (start), Ende (stop) und Inhalt (content)
(defstruct ap start stop content)

;;; Definition eines Tageseintrags aus
;;; Datum (date) und Terminliste (apl)
(defstruct day date apl)
```

***Bild 15** Definition der Strukturen für den Terminplaner*

```
(defun next-day
      (date)
 (let ((nwd (get (date-wd date)
                 'next-day))
       (mon (date-mon date))
       (day (date-day date)))
      (cond ; ? letzter Tag des Monats ->
            ;  neuer  Monat
            ((= day (get mon 'day-nr))
             (make-date :wd nwd :day 1
                 :mon (get mon 'next-mon)))
            (t (make-date :wd nwd
                   :day (+ day 1)
                   :mon mon)))))
```

***Bild 16** Definition der Funktion Next-day*

durch Angabe bestimmter Schlüsselwörter den Namen der Konstruktor- und Selektorfunktionen festzulegen, Slots von anderen Strukturen zu erben oder in gewissen Grenzen die physische Repräsentation der Struktur zu bestimmen. Für eine weitergehende Beschreibung sei auf die Sprachdefinition von Common Lisp verwiesen.

In Bild 15 ist nun dargestellt, wie die Strukturen für den Terminplaner definiert werden können.

Als nächstes soll die Funktion Next-day definiert werden, die zu einem gegebenen Datum den Nachfolger berechnet. Dazu müssen die Nachfolger der einzelnen Wochentage, der Monate und die Zahl der Tage im Monat bekannt sein.

Um dieses Wissen abzuspeichern, wird eine weitere Eigenschaft der Symbole ausgenutzt. Neben Werten und einer funktionalen Bedeutung kann man ihnen auch beliebige andere Eigenschaften zuordnen. Um diese Eigenschaften aufnehmen zu können, besitzt jedes Symbol eine sogenannte Eigenschaftsliste property list oder P-Liste, in der die Eigenschaften als Paare von *Indikator* und *Wert* abgespeichert werden können. Als Indikatoren dürfen nur Symbole verwendet werden; als Werte jedoch können beliebige Lisp-Objekte dienen. Der Zugriff zu einer bestimmten Eigenschaft erfolgt über die Funktion Get mit der folgenden Syntax:

(Get <symbol> <indikator>**)**

Mit (**setf (get** <symbol> <indikator>) <wert>) kann dem Symbol eine Eigenschaft zugeordnet werden.

Unter den Indikator Next-day wollen wir dem Symbol Sonntag das Symbol Montag zuordnen:

(setf (get 'sonntag 'next-day) 'montag)

Für die anderen Wochentage sollen analoge Zuweisungen gelten.

Übung 5: Legen Sie in den P-Listen der Monatsnamen unter den Indikatoren Next-mon und Day-nr die Namen der Folgemonate bzw. die Anzahl der Tage des entsprechenden Monats ab.

```
(defun make-calendar
       (person wd-start day-start
       month-start day-stop month-stop)
  (setf (get person 'calendar)
     (make-calendar-aux day-stop month-stop
        (make-date :wd wd-start
           :day day-start :mon month-start)
           nil))))

(defun make-calendar-aux
       (day-stop month-stop date acc)
  (cond ; Kalender fertig ?-> Umdrehen
        ; des Akkumulatorinhalts
        ((and (eq (date-mon date) month-stop)
              (= (date-day date) day-stop))
         (reverse
            (cons (make-day :date date) acc)))
        ; Restrekursion
        (t (make-calendar-aux day stop
              month-stop (next-day date)
              (cons (make-day :date date)
                 acc)))))
```

***Bild 17** Definition der Funktion Make-calendar*

```
(defun free-time
       (start stop apl)
  (cond  ;apl leer, dann Termin frei
        ((null apl)
         (list (make-ap :start start
                        :stop stop)))
        ;falsche Parameter
        ((t>= start stop) nil)
        ;Start liegt hinter erstem Termin
        ;aus apl
        ((t>= start (ap-start (car apl)))
         (free-time (if (t>= start
                           (ap-stop
                              (car apl)))
                        start
                        (ap-stop (car apl)))
                    stop (cdr apl)))
        ;Start liegt vor erstem Termin in apl
        ((t>= (ap start (car apl)) stop)
         (list (make-ap :start start
                 :stop stop)))
        ;Rekursion
        (t (cons (make-ap :start :start
                     :stop (ap-start (car apl)))
              (free-time (ap-stop (car apl))
                     stop (cdr apl))))))
```

***Bild 18** Definition der Funktion Free-time*

Mit den in den P-Listen abgelegten Informationen kann nun die Funktion Next-day definiert werden (Bild 16).

Die Funktion Make-calendar (Bild 17) legt einen leeren Terminplan für einen gegebenen Zeitraum unter dem Indikator Calendar in der P-Liste des Eigentümers ab. Da die meisten Lisp-Systeme auf 16-Bit-Rechnern nicht über Funktionen zur Berechnung des Datums verfügen, muß neben Anfangs- und Endtag und dem Monat auch noch der Wochentag des Anfangsdatums angegeben werden.

Aufgabe 7: Definieren Sie die Funktion Find-day-entry. Die Funktion soll 3 Parameter, den Tag, den Monat und die Kalenderstruktur haben und den entsprechenden Tageseintrag oder NIL liefern.

Zum Finden eines gemeinsamen Termins muß in den Terminplänen aller beteiligten Personen gesucht werden. Zunächst erzeugen wir eine Liste von freien Terminen pro Person, und dann wird der gemeinsame Termin aus dem Durchschnitt der entsprechenden Listen ermittelt. Die Liste der freien Termine wird durch die Funktion Free-Time (Bild 18) gebildet. Die Definition bedarf keiner besonderen Erklärung, nach dem Test von Grenzfällen wird die Liste rekursiv analysiert.

```
(defun common-free
       (apl1 apl2)
  (cond ((null apl1) nil)
        ((null apl2) nil)
        ;Termin in apl1 liegt vor Termin
        ;in apl2
        ((t< (ap-start (car apl2))
             (ap-start (car apl1)))
         (common-free apl2 apl1))
        ;Stop apl1 vor Start apl2 ->
        ;reduziere apl1
        ((t< (ap stop (car apl2))
             (ap-start (car apl1)))
         (common-free (cdr apl1) apl2))
        ;Stop von apl2 vor Stop apl1 ->
        ;(car apl2) ist frei
        ((t< (ap-stop (car apl2))
             (ap-stop (car apl1)))
         (cons (car apl2)
               (common-free apl1
                  (cdr apl2))))
        ;Überlappung
        (t (cons (make-ap :start (ap-start
                                  (car apl2))
                          :stop (ap-stop
                                  (car apl1)))
                 (common-free (cdr apl1)
                              apl2)))))
```

***Bild 19** Definition der Funktion Common-free*

```
(defun common-time
       (p1 p2 day month dur begin end)
  (common-time-aux
     (common-free
        (free-time begin end
           (day-apl
             (find-day-entry day
                month
                (get p1 'calendar))))
        (free-time begin end
           (day-apl
             (find-day-entry day
                month
                (get p2 'calendar)))))
     dur nil))

(defun common-time-aux
       (apl dur acc)
  (cond ((null apl) (reverse acc))
        ((>= (duration (car apl)) dur)
         (common-time-aux
           (cdr apl) dur (cons
                            (car apl) acc)))
        (t (common-time-aux
             (cdr apl) dur acc))))
```

***Bild 20** Definition der Funktion Common-time*

Aufgabe 8: Definieren Sie die Prädikate T>= und T<, die die Erfüllung der entsprechenden Ordnungsrelationen für zwei Zeitstrukturen T1 und T2 testen.

Die Funktion Common-free (Bild 19) erzeugt die Liste der gemeinsam freien Termine. Die Definition sichert, daß der erste freie Termin in APL1 vor dem ersten freien Termin in APL2 beginnt. Dann werden drei Standardfälle abgetestet: das erste Element von APL2 liegt hinter dem ersten Element von APL1, das erste Element von APL2 liegt vollständig im ersten Element von APL1, und die Elemente überlappen sich teilweise.

Jetzt fehlt nur noch eine Steuerfunktion zum Aufruf von Common-free, die die Terminpläne zweier Personen bereitstellt und die gemeinsame Liste freier Termine nach Eintragungen durchsucht, die größer als die in der Variablen *DUR* angegebene Dauer in Minuten sind (Bild 20). Die eigentliche Arbeit wird dabei von der restrekursiven Funktion Common-time-aux geleistet.

wird fortgesetzt

MASK Entwurf und Ablaufsteuerung intelligenter Dialogsysteme

- für kommerzielle und Realzeitanwendungen
- für UNIX und DOS
- interaktiver Entwurf von Masken und Druckausgabe
- virtueller Bildschirm, Fenster, Pull-down-Menüs
- strukturiertes Programmieren in der Maske
- Anschluß an C, COBOL, PASCAL ...
- MS-DOS: bewegte Symbole
  Hintergrundgrafik
  Fonteditor
- Mausunterstützung

neu vom Chemnitz-Erlanger Systemhaus!

Fordern Sie unsere kostenlose Demoversion für DOS an!

❏ 360K ❏ 1.2MB ❏ mit Handbuch 95.- M

SIGMA GmbH Dimitroffstr. 66, 9072 Karl-Marx-Stadt, 0071/44425

STRAHLUNGS-HEIZUNGEN

sichern Ihnen und Ihrer Computertechnik die Gesundheit

Beratung L. Müller
Vertrieb Brackestr. 32
Service 7065 Leipzig

Achtung Betriebe!

Verkaufen aus Überplanbeständen zu stark reduzierten Preisen elektronische Bauelemente. (Übersichtsliste bitte abfordern)

Pumpen- und Verdichteranlagen
Halle-Leipzig
Bugenhagenstraße 13–16
Halle, 4070

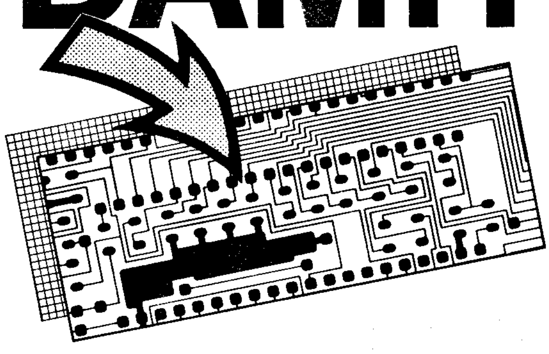
Paralleldrucker VT 23000, 23600 und 23900 an A 7150, EC 1834, XT- und AT-Kompatible

Wir haben eine Hardwarelösung entwickelt, die den Anschluß des Paralleldruckers VT 23xxx mit Interface IFSP an alle PCs mit parallelem Interface Centronics ermöglicht. Zum Betrieb sind keine zusätzlichen Softwarekomponenten erforderlich. Die elektronische Anpassung des IFSP an die Centronicsschnittstelle ist in der druckerseitigen Griffschale des Interfacekabels untergebracht und erfordert daher, außer dem von uns gefertigten Kabel, keinen weiteren Aufwand.

Wir bieten diesen Interfacekonverter als Komplettlösung ohne zusätzliches Netzteil an.

Interessenten wenden sich bitte an die:
Maschinenfabrik Nema
Abt. DW – Thurau, Fritz-Reuter-Straße
Netzschkau, 9804, Tel.: Reichenbach 49 27 02

DAMIT



IHRE KONTAKTE JETZT NOCH BESSER WERDEN

Ein Baden-Württembergisches Unternehmen – jetzt ständig mit **BÜRO IN LEIPZIG** – freut sich auf gute, neue Kontakte.

Unsere Angebotsschwerpunkte sind **Leiterplatten** mit einseitigen, doppelseitig-durchkontaktierten und Multilayer-Platinen, auch in ausgereifter SMD-Technologie sowie **elektromechanische Bauelemente,** jeweils unter Einsatz modernster Verfahrenstechniken gefertigt und durch optimale Qualitätskontrollen abgesichert.
Neben Laborkarten, Steckverbindern und Anschlußklemmen, Lampenfassungen, Schiebe-, Tast- und Wippschaltern, Löt- und Relaisleisten und Sicherungshaltern liegt unsere Angebotsstärke in anwendungsorientierten, individuell auf Ihre Wünsche zugeschnittenen Lösungen.
Wir sind leistungsstark, sehr flexibel und sehr schnell, z.B. 2 Wochen vom Layout bis zur fertigen Platine.
Haben Sie weitere Fragen?
Besuchen Sie uns doch mal oder fordern Sie ausführliches Infomaterial an.

Deutschlaender

Deutschlaender Electronic GmbH
DDR-7010 Leipzig · Kreuzstraße 13
Tel. 09 41/6 06 50 · Fax 09 41/6 00 81 · Telex 512 546

Deutschlaender Electronic GmbH
BRD-6924 Neckarbischofsheim · Bahnhofstraße 20
Tel. 072 63/63 11 · Fax 072 63/6 45 77 · Telex 782 318

SILBERMANN UND MERZ, Ludwigsburg

Wegbereiter der Informatik

NORBERT WIENER

* 1894 Columbia (Missouri)
† 1964 Stockholm

Norbert Wiener ist ein Sohn des bedeutenden jüdischen Slawisten und Literaturhistorikers Leo Wiener (1862–1939), der an der Harvard-Universität Cambridge Professor war und aus dem vom zaristischen Rußland besetzten Teil Polens stammte. Die Mutter ist aus dem Rheinland gebürtig.
Aufgrund seiner außerordentlichen Begabung begann Wiener schon mit 14 Jahren, Biologie zu studieren; nach ergänzenden Studien in Philosophie und Mathematik promovierte er bereits als 18jähriger an der Harvard-Universität mit einem Thema aus der mathematischen Logik. Anschließend setzte er bis 1915 seine Studien in Cambridge sowie in Göttingen bei dem bedeutenden Mathematiker David Hilbert (1862–1943) fort. Seine Bewunderung für Hilbert lenkte ihn schon damals auf sein künftiges Forschungsgebiet, die Anwendung der Mathematik auf physikalische und technische Probleme.
Nach seiner Rückkehr in die USA lehrte Wiener jeweils kurzzeitig an fünf verschiedenen Universitäten Logik und Mathematik (1915–1917) und versuchte sich auch als Reporter (1917–1919). Im Jahre 1920 wurde er am Institut für Technologie in Massachusetts (MIT), einem der bedeutendsten Hochschulinstitute in den USA, fest angestellt und erhielt dort 1932 eine ordentliche Professur.
Er unternahm noch mehrere Europareisen, die zum Zusammentreffen mit B. Russel, L. Landau u. a. führten; übrigens beherrschte er zehn Sprachen.
Seine Hauptarbeitsgebiete als Mathematiker waren die mathematische Analysis und die Wahrscheinlichkeitsrechnung. So entwickelte er in der Theorie der Fourier-Transformation eine harmonische Analyse für Funktionen einer komplexen Veränderlichen (1930). In der Wahrscheinlichkeitsrechnung untersuchte er eine wichtige Klasse von Zufallsprozessen, die in der Folgezeit nach ihm benannt wurden *(Wiener-Prozesse)*. Er schuf eine Interpolations- und eine Extrapolationstheorie stationärer Zufallsprozesse und hat auch Beiträge zur mathematischen Logik und zur theoretischen Physik (Potentialtheorie) geleistet.
Während des Zweiten Weltkrieges entwickelte Wiener die Theorie der Optimalfilter, mit deren Hilfe es möglich wurde, störende Nebengeräusche bei einer Nachrichtenübermittlung – zum Beispiel durch Radar – zu minimalisieren. Noch heute ist diese Filtertheorie für die Nachrichtentechnik von Bedeutung. In dieser Zeit befaßte er sich auch mit Problemen der Rechentechnik. Das Studium von Analogien zwischen Prozessen, die sowohl in elektrischen Netzwerken als auch in lebenden Organismen ablaufen, führte ihn dann schließlich zur Kybernetik, die ja dann auf die gesamte Entwicklung der Informatik entscheidende Einflüsse ausgeübt hat.
Mit seinem Buch „Cybernetics – or control and communication in the animal and the machine" gilt Wiener als Begründer der *Kybernetik*. Der Begriff Kybernetik als solcher wurde jedoch erstmals von dem französischen Physiker A. Ampère als Bezeichnung der „Wissenschaft vom Beherrschen" geprägt (1834). Wiener definiert die Kybernetik wörtlich als „Wissenschaft von der Steuerung und von den Kopplungen, ganz gleich, ob es sich dabei um Maschinen oder um lebende Organismen handelt".
Ein berühmter Schüler Wieners war der Begründer der mathematischen Informationstheorie, Claude E. Shannon, der 1940 am MIT in Mathematik promovierte. In seiner Autobiographie „I am a Mathematician" beschreibt Wiener ihn als einen Wegbereiter für die Anwendung der formalen Logik in der Schaltkreistheorie und als einen wesentlichen Promotor in der amerikanischen Entwicklung der elektronischen Rechenautomaten.

Dr. Klaus Biener

Die Hardwareinterrupts beim A 7150

Dirk Blavius, Andreas Senst, Zittau

Durch die Nutzung von Hardwareinterrupts ist eine effektive Programmgestaltung möglich. Meist werden jedoch die auch von höheren Programmiersprachen angebotenen Möglichkeiten zur Einbindung derartiger Interrupts nicht genutzt, da dazu detaillierte Kenntnisse der Hardware und des Betriebssystems des Rechners notwendig sind. Im folgenden wird gezeigt, wie der Interruptbetrieb der Schnittstellen auf der ASP-Karte (Anschlußsteuerung seriell/parallel) des A 7150 unter dem Betriebssystem DCP 1700 ermöglicht wird. Anhand eines Programmbeispieles wird die Initialisierung und Behandlung derartiger Hardwareinterrupts in Turbo-Pascal (ab Version 4.0) gezeigt.

Interruptbehandlung im A 7150

Der im A 7150 eingesetzte – 8086-kompatible – Prozessor K 1810 WM86 verfügt über ein leistungsfähiges Interruptsystem. Den maximal 256 verschiedenen Interrupttypen sind in einer Interruptvektortabelle (Speicherbereich 00000:03FFFH) jeweils 4 Byte zugeordnet. Hier werden das Codesegment und der Offset der Startadresse der Interruptbehandlungsroutine eingetragen. Beim Erkennen einer Interruptanforderung durch den Prozessor wird der aktuelle Systemzustand eingefroren. Der Interrupttyp wird ermittelt und mit 4 multipliziert. Danach zeigt dieser auf die Startadresse der jeweiligen Behandlungsroutine in der Interruptvektortabelle /1/, /2/.

Vom Interruptsystem werden prozessorinterne, programmgesteuerte und hardwaregesteuerte Interrupts unterschieden. Hardwaregesteuerte Interrupts können als maskierbare (über das Signal INTR) oder als nichtmaskierbare Anforderungen (NMI) auftreten. Die Bearbeitung und die Bereitstellung des Interrupttypes bei maskierbaren Interrupts übernimmt ein programmierbarer Interruptcontroller (PIC), der Intel-Schaltkreis 8259A.

Wenn eine Interruptanforderung an einem der 8 Eingänge (INTR1 ... INTR8) anliegt, der Interrupt dieses Eingangs erlaubt (nicht maskiert) ist und dieser die höchste Priorität im PIC besitzt, wird dem Prozessor die Nummer des Interrupttypes übermittelt. Der Typ ergibt sich aus der Basisadresse des PICs plus der Eingangsnummer.

In der ebenfalls möglichen kaskadierten Betriebsart kann an jedem der Eingänge ein weiterer PIC (Slave-PIC) angeschlossen sein, so daß maximal 64 externe Interruptquellen bearbeitet werden können. Den Interrupttyp stellt in dieser Betriebsart der Slave-PIC zur Verfügung /3/, /4/.

Das DOS-kompatible Betriebssystem DCP 1700 nutzt (im Gegensatz zum BOS 1810) die Ressourcen des A 7150 nicht voll aus, da bei Systemanlauf – durch eine entsprechende Programmierung des PICs – als Hardwareinterruptquellen nur Koprozessor-, Tastatur-, Timer- und Disketteninterrupts gestattet sind. Weitere Interrupts, zum Beispiel von der ASP, werden damit nicht angenommen.

Der A 7150 verfügt mit dem ASP-Modul über leistungsstarke Schnittstellen für die Kommunikation mit peripheren Geräten (V.24, IFFS, IFSP) sowie über zwei freie Zeitgeberkanäle für die individuelle Nutzung. Die ASP bildet durch die Hardware einen Slave-PIC nach, der seine Interruptanforderungen beim Master-PIC auf Eingang 3 (INTR3) einspeist. Damit können die auf der ASP eingesetzten 8-Bit-Schaltkreise U 855 (PIO), U 856 (SIO) und U 857 (CTC) mit Hilfe einer speziellen Adreßzuweisung in das Interruptsystem des A 7150 eingebunden werden /5/.

Generell sind bei der Nutzung von Hardwareinterrupts vom Programmierer die folgenden Probleme zu lösen:

– Programmierung der interrupterzeugenden peripheren Schaltkreise
– Programmierung des (bzw. der) Interruptcontroller
– Setzen der Interruptvektoren.

Programmierung der ASP-Schaltkreise

Zur Nachbildung des Slave-PICs durch die ASP werden die Interruptvektoren genutzt, die im Falle einer Interrupterkennung durch SIO, PIO oder CTC ausgesendet werden. Durch Wickelbrücken sind auf der ASP die Interruptcodebits 7 und 6 (Bit 7 = Low, Bit 6 = High) voreingestellt. Die oberen 6 Bit der von den Schaltkreisen ausgegebenen Vektoren werden an die Codebits 5 bis 0 geführt (siehe Tafel 1).

Tafel 1 Interrupttyperzeugung durch die ASP

	7	6	5	4	3	2	1	0	Bit
Interruptvektor des Schaltkreises (Beispiel CTC)	1	0	0	0	0	0	0	0	= 80H

	7	6	5	4	3	2	1	0	Bit
Interrupttyp	0	1	1	0	0	0	0	0	= 60H

Bei der Initialisierung der Schaltkreise mit den in Tafel 2 aufgeführten Interruptvektoren ergeben sich solche Interrupttypen, die in DCP 1700 als frei für Benutzersoftware-Interrupts aufgeführt sind. Mit der gewählten Hardwarelösung der ASP sind jedoch auch Einschränkungen beim Interruptbetrieb der Schaltkreise verbunden. Die CTC-Kanäle 1 und 2 sind nicht interruptfähig, sie werden zur Takterzeugung der V.24- bzw. der IFFS-Schnittstelle genutzt. Im Falle eines Fehlers in der Übertragung (spezielle Empfangsbedingungen oder Extern/Statusänderung) erzeugt die SIO einen Interruptvektor, der jedoch zu den selben Interrupttypen wie ein Interrupt bei leerem Sendepuffer oder bei einem empfangenen Zeichen führt. Dies muß durch den Programmierer in der Interruptbehandlungsroutine beachtet werden.

Durch Wickelbrücken sind die Interruptprioritäten auf der ASP in der Reihenfolge SIO – CTC – PIO voreingestellt. Die weitere Programmierung der peripheren Schaltkreise ist in /6/ ausführlich beschrieben, die dazu benötigten E/A-Adressen sind in Tafel 3 aufgeführt.

Eine weitere Besonderheit der Hardware ist die Interruptrückkehr für die pheripheren ASP-Schaltkreise. In U 880-Systemen überwachen diese den Datenbus und dekodieren das Ende der Interruptroutine aus dem vom Prozessor gesendeten RETI (ED4DH). Beim A 7150 muß zu diesem Zweck eine spezielle E/A-Adresse genutzt werden. Dabei wird durch die Ausgabe beliebiger Daten auf die Adresse 302H ein RETI für ASP-Schaltkreise erzeugt.

Programmierung des PICs

Die in Tafel 4 aufgeführten Steuerwörter für die Programmierung des *Master*-PICs ermöglichen den Interruptbetrieb der ASP. Die mit ICWn bezeichneten Initialisierungssteuerwörter sind nur in der Initialisierungsphase zulässig, die Ausgabe von Operationssteuerwörtern (OCWn) können jederzeit die Programmabarbeitung beeinflussen. Die exakten Bedeutungen der Steuerwörter sind in /2/ und /3/ aufgeführt.

Die Interruptrückkehr des Slave-PICs wird durch die Ausgabe beliebiger Daten auf die Adresse 300H erreicht.

Setzen der Interruptvektoren

Die Interruptvektoren sollten niemals direkt verändert werden. Das Betriebssystem DCP 1700 stellt dazu als DCP-Funktionsaufrufe den Softwareinterrupt 21H (AH:=25H – setzen Interruptvektor, AH:=35H – lesen Interruptvektor) zur Verfügung. In höheren Programmiersprachen können dafür meist spezielle Prozeduren genutzt werden.

Einbindung von Hardwareinterrupts in Turbo-Pascal

Interruptprozeduren sind in Turbo-Pascal standardisiert und besitzen die folgende Syntax:

```
procedure <NAME> (Flags,CS,IP,AX,
BX,CX,DX,SI,DI,DS,BP:WORD);
interrupt;
.
.
.
```

Tafel 2 Interrupttypen durch Slave-PIC- Nachbildung der ASP

Schaltkreis/Kanal	Initialisierung durch Interrupt-Vektor	Interrupttyp
CTC	80H	
Zeitkanal 0		60H
Zeitkanal 3		61H
PIO		
Port A (IFSP Status)	88H	62H
Port B (IFSP Daten)	8CH	63H
SIO	90H	
Port B (IFSS Senden)		64H
(IFSS Empfang)		65H
Port A (V.24 Senden)		66H
(V.24 Empfang)		67H

Tafel 3 Ausgewählte E/A-Adressen beim A 7150

Adresse	Bedeutung
C0H	PIC-Adresse (mit A0 = 0)
C2H	PIC-Adresse (mit A0 = 1)
300H	Interrupterlaubnis für ASP
302H	RETI für ASP-Schaltkreise
308H	PIO Daten Port A (IFSP Status)
30AH	Daten Port B (IFSP Daten)
30CH	PIO Steuerwort Port A
30EH	Steuerwort Port B
310H	CTC Kanal 0 (frei)
312H	Kanal 1 (Bitrate V.24)
314H	Kanal 2 (Bitrate IFSS)
316H	Kanal 3 (frei)
318H	SIO Daten Port A (V.24)
31AH	Daten Port B (IFSS)
31CH	SIO Steuerwort Port A
31DH	Steuerwort Port B

```
program interrupt_test;
{ Interrupttest CTC Kanal 0 }

uses dos,crt;

var   i          : integer;
      IVEC_ALT : pointer;

const PIC0        = $c0;   PIC2        = $c2;  { E/A Adressen}
      EI_ASP      = $300;  RETI_ASP    = $302;
      CTC_KANAL0 = $310;  CTC_KANAL3 = $316;

{-----Interruptbehandlungsroutine CTC Kanal 0--------------}
procedure ctc0(Flags,CS,IP,AX,BX,CX,DX,SI,DI,DS,ES,BP:word);
interrupt;
begin
  inc(i);                       { erhoehen Zaehler          }
  if i=MaxInt then i:=0;
  port[PIC0]     :=$20;         { Interruptrueckkehr PIC    }
  port[EI_ASP]   :=$1;          { Interrupterlaubnis ASP    }
  port[RETI_ASP]:=$1;           { RETI fuer CTC             }
end;

{-----INITIALISIERUNG-------------------------------------}
procedure init;
begin

  {--- Retten und Setzen der Interruptvektoren ------------}
  getintvec($60,IVEC_ALT);
  setintvec($60,@ctc0);

  {--- Programmierung PIC ---------------------------------}
  port[PIC0]:=$11;  { ICW1 - pegelgesteuert,               }
                    {      - kask.PIC, ICW4 erf.           }
  port[PIC2]:=$8;   { ICW2 - Basisadr. PIC im DCP          }
  port[PIC2]:=$8;   { ICW3 - Slave-PIC an INTR3            }
  port[PIC2]:=$f;   { ICW4 - keine Versch. in Slave,       }
                    {      - Puffermode, Master, auto EOI  }
  port[PIC2]:=$90;  { OCW1 - PIC Maske (EI ASP)            }

  {--- CTC0 Initialisierung ------------------------------}
  port[CTC_KANAL0]:=$80;   { Interruptvektor               }
  port[CTC_KANAL0]:=$a7;   { Interrupt, VT=256             }
  port[CTC_KANAL0]:=$ff;   { Zeitkonstante                 }

  {--- ASP Initialisierung -------------------------------}
  port[RETI_ASP]:=$1;      { EI fuer ASP                   }
end;
{----- Hauptprogramm ------------------------------------}

begin
  clrscr;  i:=1; init;
  writeln('Interrupttest CTC Kanal 0');  writeln;
  writeln('Zaehler Kanal 0 =');

  repeat
    gotoxy(20,3);write(i);
  until keypressed;


  {--- Herstellung des Ausgangszustandes------------------}
  port[CTC_KANAL0]:=$7;      { kein Interrupt von CTC     }

  port[PIC0]:=$13;  { ICW1 - pegelgesteuert, Einzel-PIC,   }
                    {      - ICW4 erf.                     }
  port[PIC2]:=$8;   { ICW2 - Basisadr. PIC im DCP          }
  port[PIC2]:=$d;   { ICW4 - keine Verschacht.,            }
                    {      - kein Puffermode,              }
                    {      - Mastermode                    }
                    {      - normal EOI                    }
  port[PIC2]:=$98;  { OCW1 - PIC Maske                     }

  setintvec($60,IVEC_ALT); { setzen der alten Interrupt-   }
                           { ,vektoren                     }
end.
```

Bild 1

Der Compiler fügt automatisch bei jeder als *interrupt* deklarierten Routine das Speichern aller Register auf dem Stack und die Freigabe dieser beim Beenden der Prozedur ein. Die Prozessorregister, die bei Prozeduraufruf als *Pseudoparameter* übergeben werden, können innerhalb der Routine verändert werden. Da diese Veränderung auf dem Stack geschieht, ist eine Parameterübergabe aus der Interruptroutine möglich.

Für die Programmierung der ASP-Schaltkreise ist die Prozedur zum direkten Zugriff auf E/A-Adressen:

port[<EA_ADRESSE>] := <STEUERWORT>

vorteilhaft. Zu beachten ist, daß sowohl <EA_ADRESSE> als auch <STEUERWORT> als *Byte* deklariert sein müssen. Mit Hilfe der beiden im Unit DOS enthaltenen Prozeduren:

GetIntVec(<I_TYP>: Byte, var <VEC_ALT>: Pointer)

und

SetIntVec(<I_TYP>: Byte, var <VEC_NEU>: Pointer)

können die Interruptvektoren problemlos ermittelt und neu gesetzt werden.

Bei der Nutzung von Hardwareinterrupts unter Turbo-Pascal sind ebenfalls Besonderheiten zu beachten. Prinzipiell sind der durch den Compiler erzeugte Maschinencode und

Tafel 4 Programmierung des PICs

Steuerwort	Ausgabe auf	Bedeutung
ICW1 = 11H	C0H	– pegelgesteuerter Interrupt – kaskadierte PICs – ICW4 erforderlich
ICW2 = 8	C2H	– Basisadresse PIC im DCP 1700
ICW3 = 8	C2H	– Slave-PIC an INTR3
ICW4 = 1DH	C2H	– volle Interruptverschachtelung bei kaskadierten PICs – Puffersteuerung – Master – normal EOI
OCW1 = 90H	C2H	– PIC-Maske (Interrupt von ASP erlaubt)
OCW1 = 98H	C2H	– PIC-Maske (Interrupt von ASP maskiert)
OCW2 = 20H	C0H	– EOI-Kommando

die Routinen zur Laufzeitbibliothek interruptfähig. Einschränkungen bestehen bezüglich der Nutzung von DOS-Funktionen und Bibliotheksfunktionen für Ein- und Ausgabe sowie für die dynamische Speicherverwaltung. Diese Routinen dürfen in der Interruptbehandlungsroutine nicht genutzt werden /7/.

Das aufgeführte Programm im Bild 1 demonstriert die Einbindung einer Interruptroutine in Turbo-Pascal 5.0 unter Nutzung des CTC-Kanals 0 der ASP. Die Nutzung anderer Interruptquellen der ASP geschieht analog.

Literatur

/1/ Bonitz, J.: Der Mikroprozessor des ESER-PC. VEB Verlag Technik, Berlin, 1989

/2/ Bäurich, H; Barthold, H.: Einführung in die 16-Bit-Mikrorechentechnik mit dem K 1810 WM86. Militärverlag, Berlin, 1988

/3/ Münzer, B.-G. u. a.: Mikroprozessorsystem K 1810 WM86. Mikroprozessortechnik, Berlin 2 (1988) 5, S. 147

/4/ Arbeitsplatzcomputer A 7150, Betriebssystem DCP 1700: Anleitung für den Systemprogrammierer. VEB Robotron-Elektronik Dresden 1/1988

/5/ Arbeitsplatzcomputer A 7150, Band 1: Rechner und Geräte, VEB Robotron-Elektronik Dresden 1987

/6/ Kieser, H.; Meder, M: Mikroprozessortechnik. VEB Verlag Technik, Berlin, 1985

/7/ HEIMSOETH & BORLAND: Turbo Pascal 4.0, Referenzhandbuch

✉ KONTAKT ☏

Technische Hochschule Zittau, Sektion K, WB Kernenergietechnik, LG Strahlenschutz und ZLFR, Theodor-Körner-Allee 16, Zittau, 8800; Tel. 6 17 24

Call for Papers

4. Fachtagung „Informatik und Schule"

WER? Gesellschaft für Informatik

WANN? 7. bis 9. Oktober 1991

WO? Oldenburg

WAS?

- Methoden, Techniken und Hilfsmittel zum Problemlösen, zum Strukturieren von Informationen, zur Kommunikation, zur Präsentation und zur Verknüpfung unterschiedlicher Medien
- Modellbildungen, Simulationen und Visualisierungen

WIE? Beiträge schicken Sie bitte bis zum 5. Januar 1991 in dreifacher Ausfertigung an das Tagungsbüro GI '91, Fachbereich Informatik, Universität Oldenburg, Postfach 2503, D-2900 Oldenburg, z. Hd. Prof. Dr. Peter Gorny. Nähere Auskünfte erteilt auch der Vorsitzende des Organisationsausschusses, Prof. Dr. Volker Claus.

Prof. Dr. Claus

8th International Symposium on Technical Diagnostics of IMEKO TC 10

September 24.–26., 1992
Berlin
Topics

- Theory an Algorithm on Technical Diagnostics
- new Measurement Devices of Technical Diagnostics
- Diagnostic for Rotating Machines
- Diagnostic Systems
- Use of Computers in Maintenance and Expert Systems

Members of the Programm Committee are from Czechoslowakia, FRG, GDR, Norway and Yogoslavia.

Please contact

Präsidium der Kammer der Technik, WGMA – TD, P.O.B. 1315, Berlin, DDR-1086; Telephone 2 26 52 18, 2 26 53 04, Telex 0114841 techkammer dd

Sturm

Fenster schaffen besseren Durchblick

Peter Wollschlaeger

Betriebssysteme gibt es relativ wenige, doch die Bedienoberflächen werden immer zahlreicher. Der Einsteiger muß sich schnell entscheiden, denn ein späterer Wechsel wird teuer. Gehört die Zukunft nun OS/2 mit dem „Presentation Manager" oder Unix mit „X-Window" oder gar der Mischung von beiden: „CXI"? Was machen die vielen anderen? (jw)

Als 1984 der Macintosh herauskam, stritt sich die Fachwelt darüber, ob man eine grafische Benutzeroberfläche (GBS) braucht oder nicht. Das Rennen ist gelaufen. Heute streiten sich die Experten nur noch darum, welche GBS die bessere ist. Somit ist zuerst eines klar: MS-DOS 3.3 pur und alle anderen kryptischen Kommandosprachen haben keine Zukunft, praktisch sind sie schon tot.

Was die Bedienoberflächen angeht, wird wohl auch einigen von ihnen das baldige Aus beschieden sein, und dabei war die Welt bis vor kurzem noch für alle so schön in Ordnung. Denn bislang gab es noch die Welt der kleinen Personalcomputer einerseits und der vernetzten Workstations andererseits. Inzwischen sind die PCs in die Workstation-Klasse hineingewachsen, Vernetzung inklusive, und das Workstation-Betriebssystem Unix mit seinen Bedienoberflächen will den PC erobern.

Um Bedienoberflächen vergleichen zu können, sollte man zuerst klären, was eine grafische Oberfläche ausmacht. Setzen Sie sich vor einen Macintosh, und Sie sehen:

- ☐ eine Maus
- ☐ Rollmenüs, die nur bei Bedarf erscheinen
- ☐ Fenster als Arbeitsflächen
- ☐ Icons (Sinnbilder) für Programme und Dokumente
- ☐ Dialogboxen, Buttons, Schieberegler und Schalter, mit denen das System über die Maus bedient wird.

Nicht alle grafischen Oberflächen sind so vollständig ausgerüstet, und die Erscheinungsformen der einzelnen Elemente sind vielfältig. Genauso groß (oder noch größer) sind die „inneren" Unterschiede. Eine GBS besteht meistens aus folgenden Komponenten:

- ☐ einem Window-System
- ☐ einem Grafik- oder Image-System
- ☐ einem API (Anwendungs-Programmierer-Interface).

Das Window-System baut die Fenster, Menüs, Dialoge und alle sichtbaren Teile auf dem Bildschirm auf. Dazu gehören: das Bewegen, Vergrößern und Verkleinern der Objekte etc.

Das Grafik- oder Image-System (Quick Draw beim Macintosh) zeichnet all diese Gebilde, aber auch Text läßt sich damit „malen". Dazu verfügt es über einen Satz von Kernroutinen (Grafik-Primitiven). Das Window-System zeichnet seine Objekte, indem es Routinen des Grafik-Systems aufruft.

Interessant ist, daß es zwei Image-Modelle gibt. Die meisten grafischen Oberflächen arbeiten mit Bit-Images, stellen also alles im Bildpunktraster dar. Das neuere Modell heißt Display-Postscript. Es sorgt speziell bei Schriften für eine bessere Übereinstimmung zwischen Bildschirm und Ausdruck, aber benötigt auch sehr viel Rechnerleistung. Der Next-Computer und die X-Window-Oberfläche arbeiten mit Display-Postscript.

Das API ist die Schnittstelle für den Anwendungsprogrammierer. Der Macintosh-Name „Toolbox" beschreibt die Funktion sehr gut. Die „Werkzeugkiste" enthält eine Sammlung von Unterprogrammen, die der Programmierer benutzen sollte. Hält er sich daran und an ein paar allgemeine Richtlinien (und greift nicht etwa direkt auf das Image-System zu), sehen in jedem Anwendungsprogramm alle Menüs, Windows und Dialoge prinzipiell gleich aus und werden auf die gleiche Art bedient. Es gibt kein System, bei dem dieses Prinzip so weit und auf so breiter Basis durchgesetzt ist, wie beim Macintosh.

Etwas andere Wege gehen Hewlett-Packard und Next. Das „New Wave" von HP ist eine Art Super-Makrosprache, mit der man aus verschiedenen Anwendungen und Objekten neue Anwendungen zusammenbauen kann. Next stellt Objekte zur Verfügung, aus denen nach den (vereinfachten) Regeln der objektorientierten Programmierung eine Applikation aufgebaut wird.

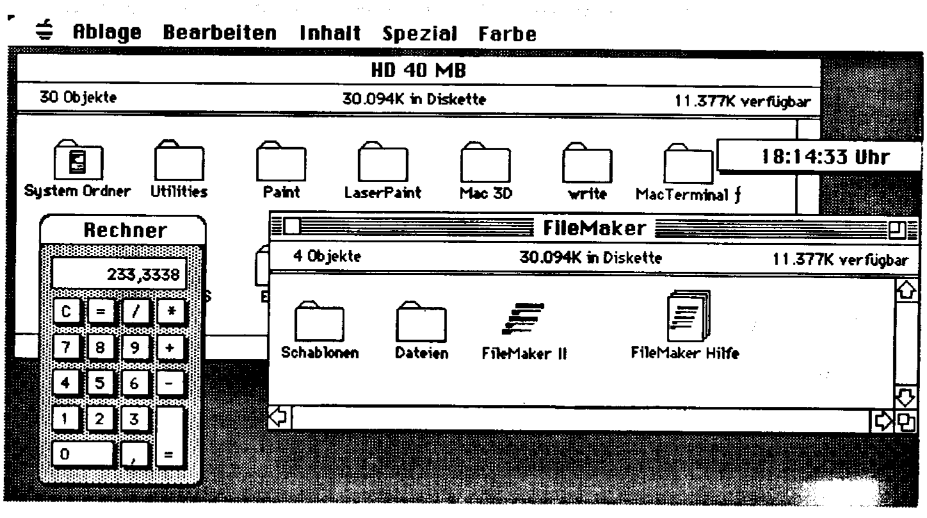
Vier Prinzipien machen eine gute grafische Oberfläche aus:

- ☐ objektorientiert
- ☐ modalfrei
- ☐ kontextorientierte Bedienerführung
- ☐ intuitive Bedienbarkeit.

Objektorientierte Programmierung (OOP) ist zwar die zukünftige (und z. T. schon gegenwärtige) Software-Technik, doch neu ist die Denkweise bestenfalls für Programmierer, die prozedural/algorithmisch arbeiten. Der Mensch denkt schon immer objektorientiert. Er hat auf seinem Schreibtisch Objekte, wie Dokumente, Ordner, einen Taschenrechner oder eine Uhr und neben dem Schreibtisch einen Papierkorb. Er schiebt mit der Maus (als verlängerte Hand) ein Dokument in den Papierkorb und tippt nicht etwa „del c:\briefe\doc.txt".

Die Darstellung dieser gewohnten Arbeitsumgebung ist eine Aufgabe einer guten grafischen Bedienoberfläche, die konsequente Umsetzung der menschlichen Arbeitsmethoden ist ein anderes Thema. Wie leicht man am zuletztgenannten Problem scheitern kann, zeigt GEM. Dort kann man zum Beispiel nicht die Dokumente beliebig auf dem Schreibtisch anordnen. Das System denkt dann völlig unlogisch, man wolle die Dokumente kopieren. Oder noch schlimmer: Es ist die natürliche Arbeitsweise, ein Dokument in den Papierkorb zu werfen und es später aus diesem wieder herauszuholen. Das gilt für den Macintosh, doch nicht für GEM. Hier haben seine Erfinder zwei Fehler gemacht. Erstens hat man ein falsches Symbol gewählt (Papierkorb anstatt Reißwolf) und zweitens hat man die „Ablage P" vergessen.

Modalfrei heißt, daß der Anwender zu jeder Zeit tun kann, was er für richtig hält und nicht etwa in einen bestimmten Modus gezwungen wird, wo das Programm sagt „nun mache das und nichts anderes". Aber auch nicht jede Aktion zu jeder Zeit ist sinnvoll oder machbar, zum Beispiel kann man in ein Dokument nichts einsetzen, wenn man vorher nichts ausgeschnitten hat. Man braucht eine kontextorientierte Bedienerführung, die nur die Aktionen sperrt, die im aktuellen Kontext

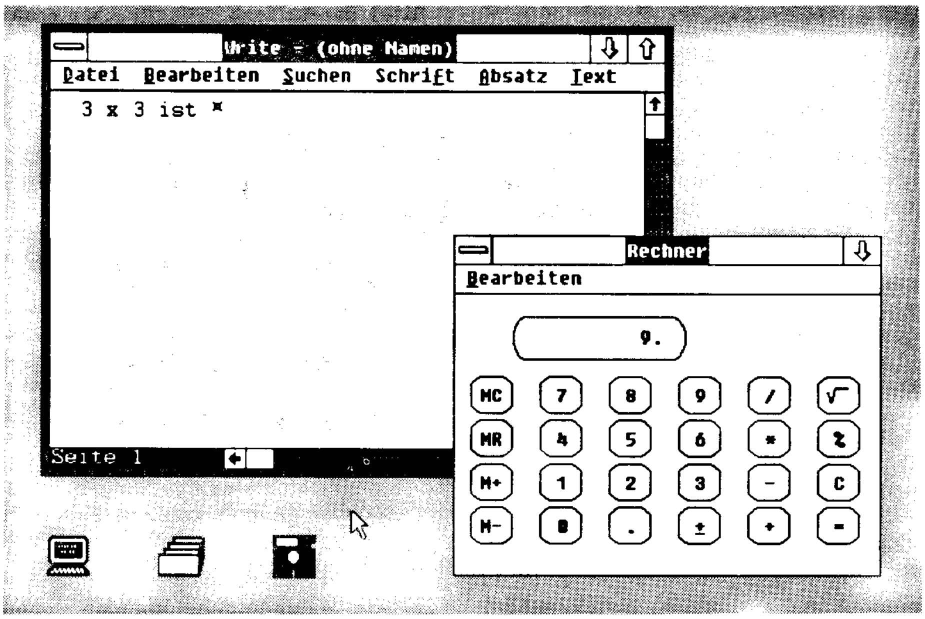


Das Vorbild: die Bedienoberfläche des Macintosh

Eine kleine Auswahl grafischer Bedienoberflächen, von Macintosh bis Sun

	Macintosh	GEM	Amiga	Windows	PM	NextStep	Sun
API	Toolbox			Windows-API	User Interface Controls	Kits	X-Lib
Window-System	Window-Manager	Application Environment System	Intuition	Graphic Device Interface	Windows-API	Window-Server	W-Window
Image-System	QuickDraw	Virtual Devic Interface	Graphics-Library	GDI-Funktionen	Graphics. API	Display-PostScript	Display-PostSript
Betriebssystem	Mac-OS	TOS	AmigaDOS	MS-DOS	OS/2	UNIX	UNIX

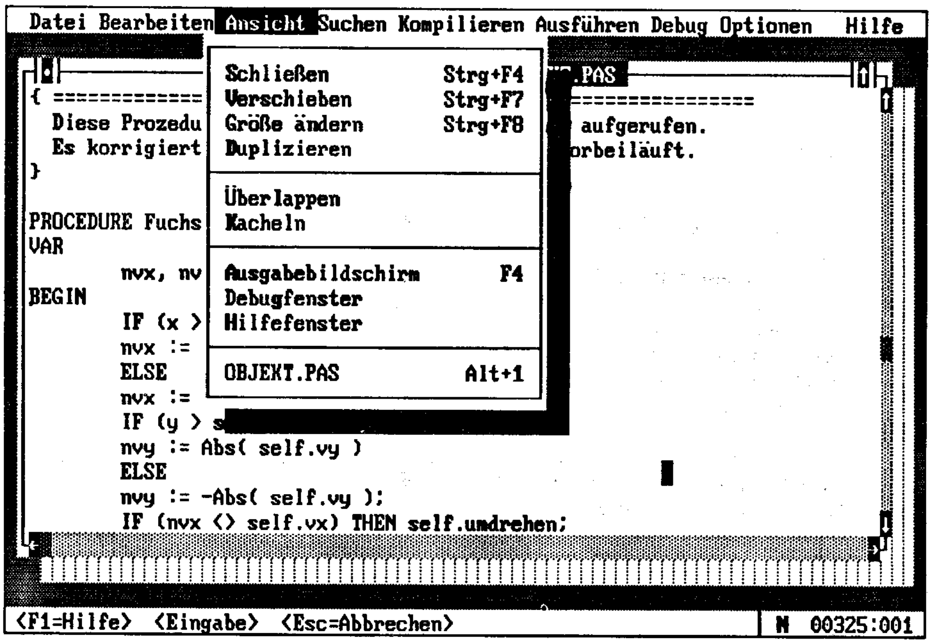
nicht zulässig sind. Das wiederum muß dargestellt werden, etwa indem man den Menüpunkt „Einsetzen" grau zeichnet.
Die intuitive Bedienbarkeit ergibt sich zwangsläufig, wenn die vorhergenannten Punkte erfüllt sind, und die grafische Darstellung der Objekte ad hoc erkennen läßt, was damit gemeint ist. Eine Grundvoraussetzung dafür sind natürlich hochauflösende Grafikbildschirme.
Nun sollte man meinen, daß das Window-System für das Erscheinungsbild und das Verhalten einer grafische Oberfläche zuständig sei, doch dem ist nicht immer so. So kann die Sun-Oberläche News (Network extensible Window System) je nach Bedarf (Anwendung) von ihrem normalen Grafik-System (Display-Postscript) auf GKS-Bit-Image umschalten, wenn die neue Task ein CAD-Programm ist. Man sieht: Nicht immer, wenn von einem Window-System die Rede ist, handelt es sich um eine grafische Bedienoberfläche.
Das berühmte „X-Window" gar ist überhaupt keine grafische Bedienerführung im oben definierten Sinne, sondern nur ein Window-System, das von verschiedenen grafischen Oberflächen benutzt wird. Das Konzept hat viele Vor- und wenige Nachteile, die Sie sofort erkennen, wenn Sie einmal über ein typisches Netzproblem nachdenken.
Wenn in einem PC-Netz eine Grafikanwendung gefahren werden soll, wird sie auf den Arbeitsplatz-Computer heruntergeladen und läuft dort unter dessen grafischer Oberfläche. Doch was machen Sie, wenn das Programm 5 MByte braucht und der Arbeitsplatzrechner nur noch 3 MByte Hauptspeicher frei hat? Sie könnten nur noch der RAM-Industrie zu Umsatz verhelfen. In der Unix-Welt macht man das eleganter: Das Programm bleibt immer auf dem Host, dieser kann eine Workstation, aber auch ein Mini oder Mainframe sein. Das Programm sendet nur Grafikanweisungen in Form sogenannter Packets an den XServer. Dieser kann eine Workstation, ein intelligentes Terminal oder auch schlicht ein PC mit XServer-Software sein. Ein Paket besteht immer aus einem Header, der unter anderem den Grafikbefehl enthält, gefolgt von den Daten. Auf dem XServer ist passend zum Window-System ein Satz sogenannter Widgets (Routinen und Objekte), installiert, der diese Packet-Befehle ausführen kann bzw. die Objekte (z. B. Elemente eines Fensters) darstellt. Es gibt auch Aktions-Widgets, die Maus- oder Tastatur-Events bearbeiten.
Das Interessante, aber auch Verwirrende an der Geschichte ist allerdings, daß jeder XServer im Netz einen anderen Widget-Satz haben kann, was bewirkt, daß ein- und dasselbe Programm auf verschiedenen Stationen unterschiedlich erscheint. Es ist deshalb durchaus denkbar, daß ein Fenster einmal eine Schließbox hat und ein andermal nicht. Der geübte Anwender wird in diesem Fall den Mauszeiger über das Window stellen, die mittlere Maustaste drücken, und siehe da, es erscheint ein Pop-Up-Menü, in dem auch „Close" steht.
X-Window wurde am MIT (Massachusetts Institute of Technology) entwickelt, mit dem Ziel, den Studenten zu ermöglichen, auf ihren Workstations zum Beispiel mit der dort installierten Textverarbeitung zu arbeiten, gleichzeitig sollten sie aber auch (in anderen Fenstern) Grafiken oder Dokumente, die irgendwo im Netz lagen, betrachten oder zusätzlich noch ein Zeichenprogramm starten



Auch unter MS-Windows läßt sich ein „Desktop", ein elektronischer Schreibtisch mit Papier und Taschenrechner, aufbauen

können. Das Ganze sollte natürlich auf dem in Unix üblichen Hardware-Mix laufen. Das Ergebnis ist ein hardwareunabhängiges, netzwerkfähiges Multitasking-Window-System. Das war 1984.
1986, der Macintosh-Erfolg machte die Konkurrenz langsam stutzig, kündigte das MIT an, daß jedermann X-Window der Version X10.4 auf Band für einen kleinen Unkostenbeitrag haben könne. Der Erfolg war umwerfend. Viele Workstation-Hersteller übernahmen X-Window, DEC und HP entwickelten sogar neue Workstations dafür. Anfang 1988 wurde es dann kommerziell. Das MIT gründete das X-Konsortium, in dem u. a. Apple, AT&T, DEC, HP, IBM, Sun und Tektronix vertreten sind. Das X-Konsortium hat die Rechte an X-Window, vergibt aber sehr großzügig Lizenzen. Das wiederum hat eine Vielfalt zur Folge, die schon an Wildwuchs grenzt. DEC hat XUI (X User Interface), HP arbeitet mit einem API namens HP X, Sun bevorzugt sein „Open Windows"-Windowsystem mit der grafischen Oberfläche „Open Look", IXL bringt X Desktop, um nur einige zu nennen.
Das bisherige Ende der Fahnenstange ist das von HP und Microsoft gemeinsam entwickelte CXI (Common X Interface), das aussieht wie der Presentation Manager, tatsächlich aber in einer X-Window-Umgebung arbeitet. Für HP ist das logisch, denn dieser Hersteller hat ja auch seine PC-Clones zu versorgen.
Damit ließ es Microsoft allerdings nicht bewenden, sondern entwickelte auch PM/X (Presentation Manager/X). Hier steht das X aber nicht mehr für X-Window, sondern für Xenix (also Unix). Anders ausgedrückt: PM/X hat ein eigenes Window-System, was zumindest laut Herstellerangabe den Vorteil haben soll, daß OS/2-Anwendungen leicht auf Unix-Systeme portiert werden können. Dennoch ist PM/X nicht der reine Presentation Manager/X, sondern eher eine Kreuzung aus CXI und Motif, womit wir wieder in der anderen Ecke wären.
Da gibt es nämlich noch die OSF (Open Software Foundation), die sich schon lange bemüht, ein einheitliches Unix zu schaffen, und nun merkte, daß die grafischen Oberflächen auseinanderlaufen. Also definierte man die neue Einheitsoberfläche „Motif".
Wir zitieren aus der Eigenwerbung: „Stellen Sie sich einmal vor, Sie gehen durch Ihr Unternehmen, vorbei an Mitarbeitern, die an Workstations, PCs oder Großrechner-Terminals sitzen. Und die Darstellungsweise ist auf allen Bildschirmen gleich, jede Maus und jede Taste funktioniert wie die andere ... Das alles ist weder Zufall noch Traum, das alles ist wahr mit der Bedienerführung OSF/Motif". Ende des Zitats.
Das haben Sie schon einmal unter dem Stichwort SAA (System-Anwendungs-Architektur) gehört? Nun, auf die kommen wir gleich. Zurück zu Motif: Das Wunderkind hat drei Väter, nämlich DECs XUI, HPs New Wave (dreidimensionale Windows) und Microsofts Presentation Manager, was das allgemeine „Look and Feel" angeht. Kaum war die Ankündigung da, versprach der Rest der Welt Unterstützung und handelte sogar.
Neben den drei Vätern wurde die SCO (Santa Cruz Operation, von der Xenix kommt) aktiv und kündigte im Februar 1988 Open Desktop an. Das ist eine Motif-Oberfläche auf 386er Unix-Basis mit DOS-Box, SQL-Datenbank und natürlich netzwerkfähig, läuft also auf Kollisionskurs mit einem zukünftigen 386er OS/2.
Die Sensation aber war die IBM-Ankündigung, Motif (eine grafische Unix-Oberfläche) zu unterstützen. Hatte IBM doch gerade die Rechte für eine andere Unix-Oberfläche erworben, nämlich das Next Step des Next-Computers. Da es nicht IBMs Art ist, zwei inkompatible Systeme gleichzeitig zu unterstützen, ist die Fachwelt nur noch am Mutmaßen. Es könnte sein, daß IBM Next Step in Reserve hält, falls sich Motif nicht durchsetzt. Denkbar ist auch, daß Next Step zum Zuge kommt, wenn der Trend mehr in Richtung Multimedia geht, und logisch wäre, wenn IBM doch noch das mächtige Toolkit von Next Step für Motif einsetzen will.
Doch nun zu SAA, der System-Anwendungs-Architektur. Genau wie die OSF für Motif hatte IBM mit SAA die große Vereinheitli-

chung angekündigt, das Ganze nur zwei Jahre früher. Doch jetzt unterstützt IBM plötzlich Motif. Ist das ein Widerspruch? Ich würde sagen, jein. Zuerst ist SAA keine grafische Bedienoberfläche, sondern nur eine Sammlung von grafischen und tastaturorientierten Benutzerschnittstellen, die nur solange einheitlich sind, wie es die Hardware am jeweiligen Arbeitsplatz zuläßt. Kann das Terminal keine Grafik, gibt es keine, im Zweifelsfall ist ein Window nur ein Stück Bildschirm ohne Rahmen.
Zum zweiten ist SAA betriebssystemübergreifend und soll alles vom PC-DOS bis zum MVS der Mainframes abdecken, Netzstandard (SNA) und Datenbankabfrage (SQL) inklusive. Alles, was in Richtung MS-Windows, Presentation-Manager oder Motif läuft, paßt rein visuell auch in das SAA-Konzept, der Anwender fühlt sich überall zu Hause, doch im Inneren gibt es keine Spur von Einheit. Jede dieser Oberflächen hat ihr eigenes Window- und Image-System und natürlich auch ein eigenes API.
SAA leidet immer noch unter der Prämisse, daß jede Anwendung auch auf simplen Uralt-Terminals laufen und ohne Maus bedienbar sein soll, sprich, daß es für alle Funktionen auch (einheitliche) Tastenkürzel geben müßte. Doch schon jetzt existieren MS-Windows-Programme, die sich den Teufel darum scheren – sie sind ohne Maus und hochauflösende Grafik einfach nicht zu gebrauchen. Der Trend zur Nur-Grafik-Bedienung wird sich fortsetzen, und so wird es zunehmend Benutzer geben, die noch nie eine tastaturorientierte Bedienoberfläche gesehen haben. Wenn eine solche Anwendung auf Ihrem Bildschirm auftaucht, können Sie, SAA hin, genormte Tastenkürzel her, damit schlicht und einfach nichts anfangen.
Betrachtet man den Rest der Welt grafischer Oberflächen, so entdeckt man dort noch den Macintosh, GEM und das „Intuition" des Amiga. GEM war als grafische Oberfläche in direkter Konkurrenz zum Mac konzipiert, mußte dann aber auf Apples Einspruch hin soweit abgerüstet werden, daß vom ursprünglichen Konzept nicht mehr viel übrigblieb. Typische GEM-Applikationen auf dem PC, Beispiel Ventura Publisher, werden deshalb häufig so installiert, daß der Anwender mit einem Batch-Kommando direkt von MS-DOS in das Programm geht und danach wieder im DOS landet. Somit fehlt die integrierende Funktion dieser grafischen Oberfläche.
Das Intuition des Amiga ist zwar Mac-ähnlicher als MS-Windows, doch gegen letzteres erhob Apple Einspruch, gegen Intuition aber nicht. Doch wahrscheinlich sahen die Apple-Strategen im Amiga keine Konkurrenz, und sie scheinen Recht zu behalten. Der Amiga ist im Home-Bereich stark, und der interessiert Apple zur Zeit offensichtlich überhaupt nicht, und wird es wohl auch nicht, solange Apple seine Macs in Tausender-Mengen an Industrie und Behörden verkaufen kann.
Das funktioniert, weil der Mac vielfach immer noch als Vorbild in puncto grafischer Bedienoberfläche gilt und eine sehr breite einheitliche Software-Basis hat. Es funktioniert sogar, obwohl diese grafische Oberfläche zu nichts kompatibel ist als zu sich selbst. Daraus können Sie schließen, daß sich viele Kunden doch am Bewährten von heute und nicht an Ankündigungen orientieren. Es zeigt aber auch, daß man Standards setzen kann, indem man einfach genug davon verkauft, besonders, wenn man dann noch gut ist.



Schon die unter MS-DOS laufenden Quicksprachen von Microsoft enthalten eine grafisch orientierte Bedienoberfläche mit Pull-Down-Menüs

Unter diesen Umständen gibt es auch in Zukunft gute Chancen für die Macintosh-Oberfläche, zumal das System inzwischen offen ist und sich mit dem „Rest der Welt" vernetzen läßt. Aber auf dem Mac läuft auch A/UX (das Apple-Unix), und Apple ist im X-Konsortium. Damit zeigt sich der andere Trend: Die Personal-Computer werden dank des Vordringens der 386er CPU salonfähig für Unix, und sogar die klassischen Workstation-Hersteller bauen Unix-Maschinen auf dieser Basis (z. B. Sun 386i). IBM und Microsoft verfolgen mit Motif und PM/X diese Richtung, doch beide werden deshalb ihr gemeinsam entwickeltes OS/2 nicht aufgeben, schlicht auch deshalb nicht, weil man jetzt erst anfängt, richtig Geld damit zu verdienen.
Bleibt das Fazit: Zukunftschancen gibt es für den Macintosh – so wie er ist oder mit Unix – und für die 386er (und Nachfolger). Ob man die Computer, einmal allein betrachtet unter dem Blickwinkel der Bedienung, unter Motif, PM/X oder mit dem original Presentation Manager fährt, ist dann eher Geschmackssache.
Was danach kommen, genauer, sich breiter durchsetzen wird, sind höher auflösende Bildschirme, plastische Grafiken, Videofilm auf und aus dem Computer sowie Musik und Sprache, kurz, Multimedia und damit schon wieder ein neuer Standard. Läßt man die Entwicklung der Personal-Computer in den letzten zehn Jahren noch einmal Revue passieren, sieht man, daß sich Standards immer sehr langsam durchsetzen. Bis ein neuer Standard anerkannt ist, wird es noch einige Zeit dauern.

Nachdruck aus Computer Persönlich, Heft 12/1990

Der DOS-Irrgarten wächst

Als allgemein bekannt kann man voraussetzen, daß MS-DOS (von Microsoft) und PC-DOS (von IBM) nahezu identisch sind. Auch die Versionsnummern stimmen überein – die letzten waren 3.3, 4.0 und 4.01. Digital Research – Entwickler des 8-Bit-Betriebssystems CP/M – bot dagegen bislang sein kostengünstigeres Konkurrenzprodukt DR-DOS 3.41 an.
Letzterer brachte nun sein DR-DOS 5.0 heraus. Es benötigt zwar mindestens 1 MByte RAM, läßt aber dem Anwender noch mindestens 620 KByte freien Speicher. Lauffähig ist DR-DOS 5.0 auf Rechnern der 80x86-Klasse. Neben Standard-Dienstprogrammen, wie einem Wordstar-kompatiblen Editor, bietet es das Speichermanagementsystem MemoryMAX, das automatische Dateitransferprogramm FileLINK, die grafische Benutzeroberfläche ViewMAX, den Festplattenbeschleuniger Disk Cache sowie für Laptops und Notebook-PCs den „Batteriesparer" BatteryMAX. Die Festplattenverwaltung ist kompatibel zu MS-DOS 4.01 und unterstützt Partitionen von bis zu 512 MByte. Anwender, die zum Beispiel bereits logische Laufwerke mit mehr als 32 MByte unter MS-DOS 4.01 eingerichtet haben, können ohne Probleme auf DR-DOS 5.0 umsteigen.
Obwohl Microsoft nie eine Version 3.4 von MS-DOS anbot, plant es jetzt neben einer neuen Version 5.0 auch eine Version 3.5. MS-DOS 3.5 soll auf die Benutzerführung der Version 4.0 verzichten und größere Festplattenpartitionen als 32 MByte zulassen. Dagegen soll MS-DOS 5.0 ähnlich wie OS/2 die Betriebssystemgrenze von 640 KByte auf 16 MByte verschieben; dabei soll der Protected Mode der Prozessoren 286, 386 und 486 genutzt werden. Auch das Dateisystem soll dem von OS/2 ähneln, und die Benutzerführung soll an MS-Windows angelehnt sein.
MP

Gründungsfieber

Teil 1

Der Traum von der Selbständigkeit

Dieter Butscher

Firmennamen tauchen im Bereich der Datenverarbeitung ebenso schnell auf, wie sie in der Versenkung verschwinden. Immer wieder wagen Mutige den Sprung in die Selbständigkeit. Was es dabei alles zu bedenken gibt, lesen Sie in einem zweiteiligen Beitrag.

Ihr Chef geht Ihnen allmählich auf die Nerven. Allmorgendlich um sieben Uhr das schrille Klingeln des Weckers. Jeden Tag die gleichen Auseinandersetzungen. Oder: Sie haben zehn Jahre lang als angestellter Programmierer gearbeitet. Reich geworden sind Sie dabei nicht. Plötzlich kommt Ihnen der zündende Gedanke. Sie sehen die Chance, mit einem selbstentwickelten Programm eine Marktlücke auszufüllen und in kurzer Zeit ein gutes Geschäft daraus machen zu können. ‚Höchste Zeit', sagen Sie, ‚sich selbständig zu machen.'

Doch leicht gedacht und schwer getan! Diese Erfahrung mußte schon mancher dynamische Gründer erleiden, wenn er sich nach langem Hin und Her entschlossen hatte, eine eigene Firma aufzubauen. Die Träume von Unabhängigkeit und dem schnellen Geld, der Freiheit von Vorgesetzten und Kollegen und das Gefühl, trotz Klingeln des Weckers die Möglichkeit zu besitzen, noch ein Stündchen länger weiterschlafen zu können, schmelzen schnell angesichts eines nervenaufreibenden Tages bei einer 7-Tage-Woche und dem immer höher werdenden Stapel der unbezahlten Rechnungen dahin. Zudem ist die Konkurrenz immer härter geworden. Die Goldgräberzeiten sind längst dahin.

Bittere Pillen

Die zündende Idee allein reicht in der Regel kaum aus, um das Gelingen des Unternehmens zu garantieren. Lange Behördengänge, schwierige Verhandlungen mit Banken, Versicherungen und Subventionsträgern, Anfragen bei der Industrie- und Handelskammer, Besuch von Gründungsmessen und -Seminaren, Termine mit Steuerberatern, Finanzierungsberatern und Rechtsanwälten gehören zum Alltag der Gründer, dazu die Entwicklung und Durchsetzung ihrer Geschäftsideen.

Diese notwendige Vorlaufphase wird oft unterschätzt. Eine falsche Entscheidung in der Gründerphase kann nicht nur zu Zeit- und Geldverlust führen, sondern das Mißlingen eines unternehmerischen Projekts – selbst wenn die Idee an sich noch so vielversprechend ist – vorprogrammieren. Die Konkursmeldungen in den Tageszeitungen zeigen, daß in der scheinbar grenzenlosen Wachstumsbranche Datenverarbeitung seit langem ein harter Konkurrenzkampf tobt, der schon manchen Firmengründer kaum das erste Geschäftsjahr überstehen ließ.

Auf geht's

Wer sich selbständig machen will, sollte bereits frühzeitig sein Augenmerk nicht nur auf die wirtschaftliche Seite des Projekts, sondern auch auf den rechtlichen und steuerlichen Aspekt richten. Hier liegen Stolpersteine verborgen, die nur durch frühzeitige Beratung und Entwicklung eines „wasserdichten" Konzepts umgangen werden können.

Wer die Idee zur Gründung eines Computerladens hat und sich trotz der gutgemeinten Ratschläge von Freunden und Bekannten immer noch nicht von der Umsetzung abhalten lassen will, muß sich als erstes überlegen, welche behördlichen Genehmigungen er benötigt.

Der An- und Verkauf von Waren, das Betreiben eines Handels- oder Dienstleistungsunternehmens ist in unserem Lande erlaubnisfrei. Dieser Grundsatz erfährt eine Vielzahl nicht unerheblicher Einschränkungen. Zunächst gilt die Gewerbefreiheit nur im Rahmen der Gesetze. Wer einen Software-Shop eröffnen und darin Raubkopien gängiger Standardprogramme zu Dumpingpreisen anbieten möchte, wird bald eine Menge Erfahrungen mit Polizei und Staatsanwaltschaft sammeln. Auf die Handels- und Gewerbefreiheit kann er sich ebensowenig berufen wie der Dealer von Rauschmitteln.

Eine Reihe von Betrieben benötigt eine spezielle Gewerbeerlaubnis. Dies gilt zum Beispiel für Gaststätten und Taxibetriebe. Hierfür erteilen die Gewerbeaufsichtsämter nach einem Prüfungsverfahren die Konzessionen.

Stellt das geplante Projekt ein Handwerk im klassischen Sinne dar, sind die Vorschriften der Handwerksordnung zu beachten. Danach darf nicht jedermann eine Goldschmiede oder eine Reparaturwerkstatt für elektronische Geräte frei eröffnen. Nur wer in der Handwerksrolle eingetragen und im Besitz der Handwerkskarte ist, darf ein Handwerk ausüben. Verstößt er gegen diese Regelung, wird er bald Schwierigkeiten mit der örtlichen Handwerkskammer bekommen, die darauf bedacht ist, die „Zunft" rein zu halten.

Der schlichte An- und Verkauf von Computer-Hard- und -Software stellt weder ein spezielles Gewerbe noch ein Handwerk dar. Liegt der Fall nicht so einfach, hilft eine Nachfrage bei der Industrie- und Handelskammer.

Gewerbefreiheit darf nicht dahingehend verstanden werden, daß jedermann ganz ohne Behördengänge ein Geschäft eröffnen kann. Wer ein Gewerbe ausübt, muß dies auf jeden Fall anmelden. Hierfür gibt es Anmeldeformulare, die auszufüllen und beim städtischen Gewerbe- oder Ordnungsamt abzugeben sind. Mit der – im übrigen gebührenpflichtigen – Gewerbeanmeldung kommt der Stein ins Rollen.

Wer meint, daß damit der Kontakt mit den Behörden und Ämtern endgültig erledigt sei, täuscht sich gewaltig. Er steht erst am Anfang des Leidensweges durch die langen Gänge der Behörden. Das Gewerbeamt leitet die Gewerbeanmeldung automatisch an verschiedene Stellen weiter. Hierzu gehören das Finanzamt, die Berufsgenossenschaft, die Industrie- und Handelskammer, das Statistische Landesamt und so weiter.

Diese Behörden übersenden dem Gründer einen ganzen Berg von amtlichen Vordrukken, Hinweisen und Formularen, mit deren Ausfüllung der Jungunternehmer sich beschäftigen darf. Hiervon sollte man sich allerdings nicht abschrecken lassen. Es handelt sich um eine notwendig Pflichtübung. Wer nämlich als Gründer die Gewerbeanmeldung unterläßt, dem droht die Verhängung eines Bußgeldes, wenn das Ordnungsamt ihm auf die Schliche kommt.

Besser haben es da schon die sogenannten Freiberufler. Die Ausübung eines freien Berufes stellt kein Gewerbe dar. Folge: Der selbständige Freiberufler kann sich den Gang zum Gewerbeamt und den damit verbundenen Papierkrieg ersparen. Nur eine Behörde kann selbst der Freiberufler nicht umgehen: das Finanzamt. Er muß seine Tätigkeit unmittelbar nach Aufnahme beim Finanzamt anzeigen und erhält von dort eine Steuernummer zugeteilt. Gegenüber dem Gewerbetreibenden hat er den entscheidenden Vorteil, daß er keine Gewerbesteuer zahlen muß. Daß im Einzelfall die selbständige Tätigkeit freiberuflicher und nicht gewerblicher Natur ist, hat ganz handfeste steuerliche und finanzielle Vorteile.

Freiberufler

Wie grenzt sich der Freiberufler vom Gewerbetreibenden ab? Die Selbständigkeit allein reicht als Merkmal nicht aus, da beide selbständig tätig sind. Im Gegensatz zum Gewerbetreibenden muß der Freiberufler – und dies ist das entscheidende Kriterium – eine höhere Tätigkeit oder Dienstleistung auf wissenschaftlichem, künstlerischem oder schriftstellerischem Gebiet ausüben. Im Paragraph 18 des Einkommenssteuergesetzes (EStG) zählt der Gesetzgeber eine Reihe typischer freiberuflicher Tätigkeiten auf. Die Aufzählung ist nicht endgültig. Wer sich in diese Riege einordnen will, dessen Berufstätigkeit muß den dort aufgeführten Berufen zumindest ähnlich sein.

Ärzten, Heilpraktikern, Dolmetschern und Übersetzern bereitet die Klassifizierung keine Schwierigkeiten. Sicher ist jedoch der Handel mit Hard- und Software kein freier Beruf, sondern ein typisches Gewerbe. Bei Tätigkeiten im Bereich der Software-Entwicklung und angrenzenden Bereichen (Beratung, Service und Wartung) wird es schon schwieriger.

Der Programmierer ist noch am leichtesten als Freiberufler einzuordnen, ähnelt seine Tätigkeit doch sehr den klassischen Berufen Dolmetscher oder Übersetzer.

Beim DV-Berater beginnt bereits die Haarspalterei. So hat der Bundesfinanzhof die Tätigkeit des Diplom-Informatikers beziehungsweise -Mathematikers als freiberuflich eingestuft (BFH, 4. 3. 83, BStBl II, S. 677). Dem „gewöhnlichen" DV-Berater ohne gehobene akademische Qualifikation blieb die Anerkennung als Freiberufler versagt (so BFH, 3. 12. 81, BStBl. II, S. 267). Im Einzelfall wird die Abgrenzung Probleme bereiten können, da viele selbständige Tätigkeiten im Bereich der DV-Branche sich nur schwer mit den klassischen freien Berufen vergleichen lassen.

Betriebsräume

Stehen für das Unternehmen geeignete Räumlichkeiten zur Auswahl, dann prüfen Sie genau die Mietverträge. Anders als beim Wohnraum ist die Gestaltungsfreiheit bei Gewerberäumen größer. Der Gewerbemietvertrag wird in der Regel über einen längeren Zeitraum (5 bis 10 Jahre), verbunden mit Staffelmieten, Preisanpassungsklauseln oder ähnlichen variablen Regelungen, fest

abgeschlossen. Über die Höhe der ortsüblichen und damit angemessenen Miete erteilt die Industrie- und Handelskammer Auskunft.
Die Anmietung von Gewerberäumen stellt immer eine erhebliche Ausgabe dar. Mancher ist daher schon auf die Idee gekommen, zunächst in der eigenen Wohnung sein Unternehmen schrittweise aufzubauen und Gewerberäume erst dann anzumieten, wenn das Geschäft bereits einigermaßen läuft. Hier ist Vorsicht geboten. In fast allen Bundesländern bestehen „Zweckraumentfremdungsverbote“, die es untersagen, Wohnungsräume für gewerbliche Zwecke zu nutzen. Wann eine solche Zweckentfremdung vorliegt, bestimmt sich freilich nach dem Umfang der Geschäfte. Der Verkauf von fünf PCs innerhalb eines Jahres in der eigenen Wohnung stellt gewiß noch keine Zweckentfremdung dar.
Erkundigen Sie sich vorsichtshalber beim Wohnungsamt, ob es sich bei ihrem Wunschobjekt um Geschäfts- oder Wohnräume handelt, die der Vermieter verbotenerweise als Gewerberäume anbietet, um eine höhere Miete zu kassieren. Mißtrauen ist angebracht, wenn Küche und Bad eher auf eine Wohnung als einen Geschäftsraum hindeuten.

Kaufleute

Sind die Geschäftsräume angemietet, frisch renoviert und der Sekt für die Betriebseröffnung kaltgestellt, sollten Sie vor Anbringung des Firmenschildes eine gewichtige Frage stellen: ‚Bin ich ein Kaufmann?'

Name der entgegennehmenden Gemeinde/Stadt | Gemeindekennzahl | GewA 1

Gewerbe-Anmeldung nach § 14 GewO oder § 55 c GewO sowie § 1 GewAnzV
Bitte mit Schreibmaschine oder in Blockschrift vollständig und gut lesbar ausfüllen sowie die zutreffenden Kästchen ankreuzen.

Angaben zum Betriebsinhaber
Bei Personengesellschaften (z. B. OHG) ist für jeden geschäftsführenden Gesellschafter ein eigener Vordruck auszufüllen.
Bei juristischen Personen (z. B. GmbH) ist bei Feld Nr. 3 bis 10 und Feld Nr. 30 und 31 der gesetzliche Vertreter anzugeben.
Die Angaben für weitere gesetzliche Vertreter zu diesen Nummern sind auf der Rückseite des Vordrucks ☐ oder einem Beiblatt ☐ oder weiteren Vordrucken ☐ gemacht.

1 Im Handels-, Genossenschafts- oder Vereinsregister eingetragener Name
2 Ort und Nr. der Eintragung
3 Familienname
4 Vornamen
5 Geburtsname (nur bei Abweichung vom Familiennamen)
6 Geburtsname der Mutter
7 Geburtsdatum
8 Geburtsort (Ort, Kreis, Land)
9 Staatsangehörigkeit deutsch ☐ andere:
10 Anschrift der Wohnung und Telefon-Nr.

Angaben zum Betrieb
11 Zahl der geschäftsführenden Gesellschafter (nur bei Personengesellschaften):
Zahl der gesetzlichen Vertreter (nur bei juristischen Personen):
12 Anschrift der Betriebsstätte und Telefon-Nr.
13 Anschrift der Hauptniederlassung und Telefon-Nr.
14 Anschrift der früheren Betriebsstätte
15 Angemeldete Tätigkeit (genau angeben: z. B. Herstellung von Möbeln, Elektroinstallationen und Elektroeinzelhandel, Großhandel mit Lebensmitteln usw.)
17 Datum des Beginns der angemeldeten Tätigkeit
18 Art des angemeldeten Betriebes Industrie ☐ Handwerk ☐ Handel ☐ Sonstiges ☐
19 Anzahl der voraussichtlich im angemeldeten Betrieb beschäftigten Arbeitnehmer:
Die Anmeldung wird erstattet für
20 einen selbständigen Betrieb ☐ eine Zweigniederlassung ☐ eine unselbständige Zweigstelle ☐
21 ein Automatenaufstellungsgewerbe ☐
22 ein Reisegewerbe ☐
Wegen
23 Neuerrichtung des Betriebes ☐
24 Übernahme eines bereits bestehenden Betriebes (z. B. durch Kauf, Pacht, Erbfolge, Änderung der Rechtsform, Gesellschaftereintritt) ☐
26 Name des früheren Betriebsinhabers (falls bekannt)

Falls der Betriebsinhaber für die angemeldete Tätigkeit eine Erlaubnis benötigt, in die Handwerksrolle einzutragen oder Ausländer ist:
28 Liegt eine Erlaubnis vor? Nein ☐ Ja, erteilt am/von (Behörde):
29 Liegt eine Handwerkskarte vor? Nein ☐ Ja, ausgestellt am/von (Handwerkskammer):
30 Liegt eine Aufenthaltserlaubnis vor? Nein ☐ Ja, erteilt am/von (Behörde):
31 Die Aufenthaltserlaubnis enthält keine Auflage oder Beschränkung ☐ enthält folgende Auflage oder Beschränkung:

Einer Übermittlung der in der Gewerbeanzeige enthaltenen Angaben über Namen (Felder 1, 3 u. 4), betriebliche Anschrift (Felder 12 u. 13) und angemeldete Tätigkeiten (Feld 15) an Dritte (z. B. Adreßbuchverlage, Versicherungen, Markt- und Meinungsforschungsinstitute usw.) zum Zwecke der Werbung oder Meinungsforschung stimme ich zu ☐ stimme ich nicht zu ☐ (Zutreffendes Kästchen bitte ankreuzen)

Bitte die Hinweise auf der Rückseite beachten.
Diese Anmeldung wird gemäß § 15 Abs. 1 GewO bescheinigt.
Verwaltungsgebühr DM ______ (Behörde)
(Dienstsiegel)
Gebührenliste Nr. ______
(Datum) (Unterschrift)

Empfangsbescheinigung für den Anzeigepflichtigen

Nachdruck mit freundlicher Genehmigung des Deutschen Gemeindeverlages, Hannover.

Den Schreibstift fest aufdrücken! Neun Durchschriftblätter für interessierte Behörden und Institutionen verbergen sich hinter dem Deckblatt der Gewerbeanmeldung.

Während der landläufige Sprachgebrauch darunter den Obsthändler an der Ecke oder den Fischverkäufer versteht, sieht der Jurist den Begriff aus einer anderen Sicht. Kaufmann ist derjenige, der eines der im Handelsgesetzbuch (Paragraph 1 HGB) aufgezählten Handelsgewerbe betreibt. Hierzu zählen unter anderem folgende Geschäfte:
- Anschaffung und Veräußerung von Waren
- Be- oder Verarbeitung von Waren für andere
- Verlagsgeschäfte und Druckereien.

Der Verkauf von Hardware stellt ein Handelsgewerbe dar, wie die Reparatur von PCs oder die Eröffnung eines Fachverlages für DV-Literatur. Von der Einordnung eines Gewerbes als *kaufmännisch* hängen rechtlich bedeutsame Konsequenzen ab.
Den Kaufmann treffen Buchführungs- und Aufbewahrungspflichten, dazu gesteigerte Sorgfaltspflichten im Rechtsverkehr. Er hat eine Firma zu führen.
Verschiedene Verbraucherschutzgesetze, darunter das Abzahlungsgesetz oder das Gesetz über Allgemeine Geschäftsbedingungen gelten für den Kaufmann nicht oder nur in eingeschränkter Form. Er kann eine Bürgschaftserklärung bereits in mündlicher Form abgeben. Die kaufmännischen Mängelrügefristen beim Erwerb fehlerhafter Waren sind bedeutend kürzer als für Nichtfachleute.
Kurz gesagt: Der Kaufmann muß im Gegensatz zum einfachen Gewerbetreibenden mehr Rechtspflichten beachten. Kaufleute verstricken sich dadurch leichter in den Maschen des Gesetzes.
Wer ein Handelsgewerbe betreibt, indem er zum Beispiel Computer verkauft und sein Geschäft nur einen geringen Umfang besitzt, ist nach dem Handelsgesetzbuch (HGB) lediglich ein sogenannter Minderkaufmann. Die Pflicht zur Buchführung und zur Führung einer Firma, sowie andere Verpflichtungen nach dem HGB, treffen den Minderkaufmann nicht.
Umgekehrt ist derjenige, welcher zwar kein Handelsgewerbe betreibt (etwa ein gewerblicher DV-Systemberater), dessen Geschäft jedoch einen erheblichen Umfang angenommen hat und der mehrere Mitarbeiter beschäftigt, als Kaufmann mit allen Rechten und Pflichten anzusehen („Sollkaufmann“). Er muß sich in das Handelsregister eintragen lassen. Wer dort aufgeführt ist, muß sich im Rechtsverkehr auf jeden Fall auch als Kaufmann behandeln lassen: Hat er eine mündliche Bürgschaft oder ein Schuldanerkenntnis abgegeben, die nur bei Kaufleuten ohne Schriftform wirksam sind, kann er sich später nicht damit herausreden, daß er gar kein Handelsgewerbe betreibt und seine Erklärungen nicht wirksam gewesen sind.
Eine der wichtigsten Vorschriften für den Kaufmann ist die Pflicht zur Führung einer Firma. Im Handelsrecht ist unter einer Firma nicht das Geschäft, sondern der Name des Geschäftes zu verstehen. Die Firma stellt eine Geschäftsbezeichnung dar, unter welcher der Kaufmann im Handelsverkehr auftritt. Der Phantasie bei der Namenswahl sind deutliche Grenzen gesetzt.
Die Firma muß zunächst den sogenannten Firmenkern und die Firmenzusätze enthalten. Bei einem Einzelunternehmen besteht der Firmenkern aus dem Familiennamen mit mindestens einem ausgeschriebenen Vornamen. Dem Kern kann ein Zusatz angehängt werden, der auf die Art des ausgeübten Ge-

schäfts hinweist. Möglich wäre also: *Fritz Müller Software Vertrieb* oder *Fritz Müller DV Beratung*.
Nicht zulässig wäre hingegen: *Müller Software* oder *Fritz Müller Software Vertriebsgesellschaft*. Im ersten Beispiel wird der Vorname unterschlagen, im zweiten das Vorliegen einer Gesellschaft vorgetäuscht. Dies ist dem Einzelunternehmer versagt.
Verboten sind ferner Zusätze, die geeignet sind, eine Täuschung oder Irreführung über Art und Umfang des Geschäftes hervorzurufen. Wer seinen im hinterbayrischen Einödshof ansässigen Handel *international DV systems* nennen will, darf mit Schwierigkeiten bei der Eintragung im Handelsregister rechnen, vorausgesetzt, er betreibt nicht tatsächlich einen weltumspannenden Handel mit Computern.
Ein Computershop, der seinen Geschäftskreis lediglich auf Wuppertal beschränkt, darf sich nicht *Computer-Versand Deutschland* nennen. Besteht am selben Ort schon ein Geschäft einer gleich oder ähnlich lautenden Firma, wird das Vorliegen einer Verwechslungsgefahr angenommen.
Über die Zulässigkeit von Firmenzusätzen entspinnen sich in der Praxis regelmäßig heftige Schriftwechsel mit den Registergerichten, welche für die Eintragung der Firmen zuständig sind. Das Registergericht – eine Abteilung des örtlichen Amtsgerichts – muß von Amts wegen die Zulässigkeit der Firma prüfen. Oft wird dabei vorschnell und aus übertriebener Sorge für das „Wohl der Allgemeinheit" eine Täuschungs- oder Verwechslungsgefahr unterstellt.
Will man dies vermeiden, besteht die Möglichkeit, vorab eine Anfrage an die Industrie- und Handelskammer zu richten. Diese teilt dann mit, ob gegen die Firmenwahl Bedenken bestehen könnten oder nicht. Sind bereits mehrere Geschäfte derselben Branche ortsansässig, hilft ein einfacher Blick in das aktuelle Branchenbuch.
Die Wahl der richtigen und zulässigen Firmenbezeichnungen stellt nicht nur Kaufleute vor Probleme. Wer ein Gewerbe betreibt, ohne Kaufmann zu sein, oder freiberuflich tätig ist, darf im Geschäftsverkehr zwar keine Firmenbezeichnung benutzen, er kann sich aber eine Geschäftsbezeichnung geben. Beispiel: *Direkt Unternehmensberatung*. Auch bei der Wahl von Geschäftsbezeichnungen ist Vorsicht am Platz. Geschäftsbezeichnungen unterliegen dem Schutz des Namensrechts!
Wer eine Geschäftsbezeichnung an bereits vorhandene Bezeichnungen von Konkurrenten anlehnt und damit von deren gutem Ruf profitieren möchte, ruft Täuschungs- oder Verwechslungsgefahr hervor. Der Konkurrent kann gegen die Bezeichnung Maßnahmen ergreifen.

Konzepte

Wer sich selbständig machen möchte, diesen Schritt jedoch nicht allein gehen will, kommt nicht umhin, sich über die Wahl der passenden Rechtsform und der Firmenbezeichnung den Kopf zu zerbrechen. Dabei spielen auf der Suche nach dem passenden „Rechtskleid" verschiedene Gesichtspunkte eine Rolle. Haftungsprobleme sind neben der vertraglichen Absicherung gegenüber dem Partner genauso zu berücksichtigen wie steuerliche Überlegungen.
Die hohe Rate von Konkursen und finanziellen Pleiten von Jungunternehmen im Bereich der DV zeigt nicht nur, daß die Goldgräberzeit längst vorbei ist. Die Erfahrung lehrt, daß die meisten Schiffbrüche sich in den ersten beiden Jahren ereignen. Überstürzte Gründungen durch unüberlegte Marktstrategien, mangelhafte Finanzierungskonzepte, Überschätzung der Absatzmöglichkeiten, Unterschätzung der realen Kostenfaktoren, unausgefeilte rechtliche und steuerliche Konzepte können hierbei das Schiff bereits vor dem Auslaufen aus dem Hafen zum Sinken bringen.
Ideale Lösungen und Patentrezepte zur Wahl der passenden Rechtsform gibt es nicht. Das Zivilrecht stellt eine Vielzahl möglicher Formen zur Verfügung. Welche ausgewählt wird, bestimmen eigene Entscheidungskriterien und die verfolgte wirtschaftliche Zielsetzung.
Spielen Finanzierungsfragen keine Rolle und wollen Sie Ihr Projekt auf eigene Faust beginnen, macht die Gründung einer Aktiengesellschaft wenig Sinn. In diesem Fall sollten Sie besser als eigenständiger Freiberufler, Einzelkaufmann, Einmann-GmbH oder gar Einmann-GmbH & Co KG verkleidet auftreten.
Sind Sie im Besitz eines vermarktbaren Know-hows oder brauchbarer Programmierkenntnisse, fehlt es Ihnen aber an dem nötigen Kleingeld zum Aufbau eines Unternehmens, kann die Einbindung eines finanzstarken Geldgebers, der an den Entscheidungen des Unternehmens nicht beteiligt ist, in Form einer Kommanditgesellschaft (KG) oder als stiller Gesellschafter sinnvoll sein.

Einzelkämpfer

Wollen Sie die Entscheidungen und das wirtschaftliche Risiko nicht allein tragen und kennen Sie einen oder mehrere vertrauenswürdige und gleichberechtigte Partner, dann ist die Gründung einer Gesellschaft bürgerlichen Rechts (GbR), einer offenen Handelsgesellschaft (OHG) oder einer Gesellschaft mit beschränkter Haftung (GmbH) zu erwägen.
Stellt das Geschäft ein hohes Risiko dar? Müssen Sie alles auf eine Karte setzen, um durchzukommen? Dann könnte die Gründung einer GmbH wenigstens davor bewahren, im Falle einer Pleite mit dem gesamten Privatvermögen zu haften.
Anhand dieser Kriterien kann bereits eine Vorauswahl getroffen werden. Die Wahl der Rechtsform allein befreit noch nicht von der Notwendigkeit, einen Gesellschaftsvertrag mit dem Partner durchzusprechen und zu formulieren. Ein Blick ins GmbH-Gesetz oder in das Handelsgesetzbuch steht dann am Anfang der Aktivitäten.
Für alle gängigen Gesellschaftsformen gibt es sogenannte Mußpunkte, die ein Gesellschaftsvertrag regelt. So fordert das GmbH-Gesetz einen oder mehrere Geschäftsführer und regelmäßige Gesellschafterversammlungen. In einem Vertrag zur Gründung einer Gesellschaft bürgerlichen Rechts ist zu regeln, wer die Gesellschaft nach außen hin vertreten darf. Im einzelnen bleibt immer noch ein weiter Raum für individuelle vertragliche Gestaltungsmöglichkeiten, denn es gilt der Grundsatz der Vertragsfreiheit.
In diesem und im nächsten Teil sollen im Überblick die wichtigsten Rechtsformen für Unternehmen dargestellt werden. Der Blick richtet sich schwerpunktmäßig auf folgende Gesichtspunkte:
– Welches Kapital ist für die Gründung der Gesellschaft erforderlich?
– Wer haftet persönlich?
– Wer hat die Entscheidungsgewalt?
– Welche Gründungsformalitäten sind erforderlich?
– Welche steuerlichen Auswirkungen hat die Gesellschaftsform?

Lonely hearts

Wer partout keinen Partner findet oder als eingefleischter Einzelkämpfer lieber ein Unternehmen auf eigene Faust gründet, dem bleibt der Weg des Freiberuflers oder die Rechtsform des Einzelkaufmanns. Ob ein Einzelunternehmen als freiberuflicher oder kaufmännischer Betrieb gilt, hängt davon ab, ob es sich um ein Gewerbe handelt. Den Unterschied und die oft schwierige Abgrenzung habe ich bereits dargestellt, so daß ich mich hier auf das kaufmännische Einzelunternehmen beschränken kann.
Der Vorteil des Einzelkämpfers liegt auf der Hand: Er kann seine Entscheidungen alleine treffen und muß sich nicht mit anderen Menschen und Meinungen herumplagen. Weiterer Vorteil: Ein bestimmtes Stammkapital ist für die Führung eines Einzelunternehmens nicht erforderlich, ebensowenig der Abschluß eines Gesellschaftsvertrages. Mit wem sollte dieser auch abgeschlossen werden? Beginnt der Einzelunternehmer sein Geschäft im bescheidenen Umfang, hält sich der bürokratische Hürdenlauf in Grenzen: Die Anmeldung im Handelsregister und die Pflicht zum Führen von Handelsbüchern entfällt. Über seine Firma muß sich der Einzelkaufmann noch keine Gedanken machen. Erforderlich ist nur die Anmeldung beim Gewerbeamt und die Anzeige beim Finanzamt.
Laufen die Geschäfte gut und wächst das Unternehmen, besteht die Verpflichtung, die

Mit acht Grundfragen sollten sich Existenzgründer von Anfang an auseinandersetzen.

- Wie soll das Unternehmen finanziert werden?
- Welche Rechtsform genießt bei Banken, Kreditgebern und öffentlichen Förderungsinstitutionen Ansehen?
- Wie sieht es mit der Haftung für Schulden und Kredite aus?
- Wer leitet das Unternehmen?
- Welche Steuerlasten kommen auf mich zu?
- Mit welchem Aufwand und welchen Formalitäten ist die Gründung des Unternehmens verbunden?
- Wer bringt das Kapital ein?
- Sollen Angestellte beschäftigt werden?

Eintragung in das örtliche Handelsregister vorzunehmen und dem Unternehmen einen Namen zu geben. Wann der Minderkaufmann sich zum Vollkaufmann und sein Geschäft sich damit zu einem in kaufmännischer Weise eingerichteten Gewerbebetrieb wandelt, bestimmt hauptsächlich der erzielte Jahresumsatz. Die Grenze zum Vollkaufmann liegt bei einem Jahresumsatz von etwa 150 000 DM bis zu 250 000 DM.

Dabei ist der Umsatz nicht das einzige, maßgebliche Kriterium. Selbst ein Kleinbetrieb darf höhere Umsätze erzielen, während das Amt umgekehrt bei geringeren Umsätzen schon annehmen kann, daß ein vollkaufmännisches Gewerbe besteht: etwa, wenn das Unternehmen einen umfangreichen Warenbestand aufweist, der eine Lagerbuchhaltung erforderlich macht.

Nachfragen

Jedenfalls sollte sich der Jungunternehmer nicht nur über seine in die Höhe schießenden Umsatzzahlen freuen, sondern sich rechtzeitig bei der Industrie- und Handelskammer erkundigen, ob sein Betrieb als vollkaufmännisch zu betrachten und er damit zu einer Eintragung in das Handelsregister verpflichtet ist.

Gleichgültig, ob das Unternehmen als Kleinbetrieb oder vollkaufmännischer Betrieb auftritt, trifft den Inhaber immer die volle persönliche Haftung für alle geschlossenen Verbindlichkeiten. Wer mit seinem Betrieb Schiffbruch erlitten hat, den Laden schließt und den Stapel der unbezahlten Rechnungen in sein Erinnerungsalbum einsortiert, wird die Gläubiger nicht zufriedenstellen. Es gilt der Grundsatz, daß der Unternehmer mit seinem gesamten Privatvermögen für alle in dem Betrieb begründeten Schulden haftet!

Der Einzelkaufmann kann sich nicht hinter dem Rechtskleid des Unternehmens verstekken und bei einer Pleite seine Gläubiger damit vertrösten, daß das Unternehmen seine Pforten geschlossen habe und es ihm schließlich auch nicht gut gehe. Gelingt es ihm nicht rechtzeitig, sich in ein ferngelegenes Land abzusetzen, werden ihm die Nachwehen des gescheiterten Unternehmens noch lange zu schaffen machen.

Trotz dieser Nachteile und wirtschaftlichen Risiken eignet sich das einzelkaufmännische Unternehmen für alle, die nicht gleich im großen Stil loslegen möchten. Läuft der Laden gut, bietet sich immer noch die Gründung einer Gesellschaft an – etwa einer GmbH unter Einbeziehung weiterer Partner oder als Einmann-GmbH.

Die steuerliche Seite sieht nicht so rosig aus. „Wenn ich erst selbständig bin, kann ich ja meinen Wagen und die Gaststättenrechnungen vom Finanzamt bezahlen lassen." Solche Sprüche gehören eher an den Stammtisch unzufriedener Angestellter, denen sich über den letzten Steuerbescheid des Finanzamtes die Haare sträuben, als in den Gründungskoffer eines Unternehmers.

Belege

Zwar kann der Unternehmer Betriebsausgaben geltend machen und hierbei einen Firmenwagen oder Aufwendungen für die Bewirtung von Kunden ansetzen. Diese werden nur von den Einnahmen abgezogen. Wer keine Einnahmen hat, kann auch nichts abziehen. Anstatt sich auf die vielfältigen Möglichkeiten steuerlicher Tricksereien zu verlegen, sollte am Anfang eher das Augenmerk auf der Einnahmenseite liegen und darauf, daß der Rubel erst einmal rollt.

Später kann man Wege suchen, dem Fiskus ein Schnippchen zu schlagen. Der Gang zum Steuerberater lohnt sich auf jeden Fall. Das Fahrzeug sollte frühzeitig in das Betriebsvermögen des Unternehmens eingebracht werden. Quittungen und Belege für alle betrieblichen Ausgaben sollten nicht im Papierkorb verschwinden, sondern in einem Ordner abgeheftet werden. Wer es allerdings zu toll mit den Betriebsausgaben treibt, darf sich auf die erste Betriebsprüfung freuen. Stellt der Prüfer bei seinem gestrengen Blick auf die Gaststättenrechnungen fest, daß gar kein Geschäftsfreund, sondern die Ehefrau bewirtet worden ist, stehen Steuernachzahlungen und unter Umständen ein Strafverfahren wegen Steuerhinterziehung ins Haus.

12 × 1500 = 30 000?

Stimmt die Einnahmenseite, ist der Wagen auf den Betrieb umgeschrieben, führt der Jungunternehmer emsig Buch über seine Einnahmen und Ausgaben, gibt er seine Belege und Quittungen regelmäßig bei seinem Steuerberater ab, erlebt er am Ende des ersten Geschäftsjahres dennoch unvermeidliche Überraschungen. Der vom Steuerberater erstellte Jahresabschluß weist einen Gewinn von 30 000 DM aus, obwohl der junge Unternehmer recht spartanisch gelebt und monatlich nur 1500 DM aus der Kasse entnommen hat. 12 × 1500 rechnet sich nach Adam Ries aber zu 18 000 DM und nicht zu 30 000 DM aus. Ein Blick auf die letzten Kontoauszüge offenbart nur ein deprimierendes Minussaldo und bringt nicht des Rätsels Lösung. Wo sind die fehlenden 12 000 DM geblieben?

„Das Zauberwort heißt Abschreibung für Abnutzung (AfA)", klärt der Steuerberater auf. Der Unternehmer hat sich bereits im ersten Geschäftsjahr eine Mehrplatz-DV-Anlage mit Software zum stolzen Preis von 48 000 DM angeschafft. Den Kauf ermöglichte ein Bankdarlehen. ‚Wenn ich für eine DV-Anlage 48 000 DM ausgebe, muß dies auf der Ausgabenseite doch voll zu Buche schlagen', mutmaßte der Unternehmer und scheute daher die Anschaffung der von der Bank finanzierten teuren DV-Anlage nicht.

So denkt der Fiskus leider nicht. Bei Anschaffung von Wirtschaftsgütern mit einem Wert von mehr als 800 DM wird der Anschaffungspreis über mehrere Jahre verteilt als Betriebsausgabe angesetzt. Das heißt: Obwohl der Anschaffungspreis bereits im ersten Jahr voll bezahlt wurde, sind nur 25 Prozent hiervon im ersten Jahr und jeweils weitere 25 Prozent in den nächsten drei Jahren als Betriebsausgabe abzusetzen.

Vom betriebswirtschaftlichen Standpunkt ist dies leicht nachvollziehbar: Würde eine größere Anschaffung bereits im Anschaffungsjahr in vollem Umfang angesetzt, hätte der Unternehmer damit im ersten Jahr hohe Betriebsausgaben und eher hohe Verluste, in den folgenden Jahren aber geringere Ausgaben und eher Gewinne. Eine gleichmäßige Verteilung über die voraussichtliche Nutzungsdauer (lineare Abschreibung) vermeidet diese Unausgewogenheit.

Der so im Jahresabschluß ermittelte Gewinn ist von dem Einzelkaufmann als Einkommen aus selbständiger Tätigkeit oder dem Gewerbebetrieb zu versteuern. Hierzu muß er eine Einkommenssteuererklärung abgeben. Geschieht dies nicht freiwillig, hilft das Finanzamt auf die Sprünge. Konnte er als Arbeitnehmer früher noch pauschale Freibeträge in seiner Steuererklärung ansetzen, entfällt diese Möglichkeit bei Einkünften aus selbständiger Tätigkeit. Hier gibt es lediglich einen Freibetrag in Höhe von 5 Prozent des zu versteuernden Gewinns, maximal 1200 DM.

Neben der Einkommenssteuer muß der gewerbetreibende Einzelunternehmer (nicht der Freiberufler) Gewerbesteuer entrichten.

Von dem im ersten Geschäftsjahr erzielten vorzeigbaren Gewinn verbleibt nach Abzug der Einkommens- und Gewerbesteuer oft nur ein lächerlicher Betrag übrig. Mancher hat deshalb schon seiner früheren Angestelltentätigkeit nachgetrauert. Daher sollte man bereits frühzeitig mit dem Steuerberater die steuerlichen Belastungen durchsprechen und einkalkulieren.

wird fortgesetzt

Der Autor ist Rechtsanwalt in Wuppertal.
Nachdruck aus c't – Magazin für Computertechnik, Hefte 5 und 6/1990

Unternehmenssimulation per Software

Ein großes Problem sowohl für Unternehmensgründer als auch für Manager ist das betriebswirtschaftliche Begreifen und Führen ihrer Firma. Auch die beste Geschäftsidee ist wertlos, wenn sie nicht konsequent und betriebswirtschaftlich fundiert umgesetzt wird. Da nur wenige in der Lage sind, alles nach Gefühl richtig zu machen, bietet die Firma FILAN-Software aus Köln mit dem Paket **Busy** ein Arbeitsmittel an, mit dem analytisches Managen auch ohne umfangreiche Betriebswirtschaftskenntnisse möglich sein soll.

Angefangen mit der Firmengründung oder einem anderen definierten Ausgangspunkt können die folgenden 4 Geschäftsjahre geplant und an Hand der entsprechenden Soll/Ist-Vergleiche die Lebensfähigkeit des Unternehmens überprüft werden. Dabei helfen die Gewinn/Verlust-Rechnung nach Produktgruppen sowie die Betrachtung einer Reihe betriebswirtschaftlicher Kennziffern. Durch Änderung bestimmter Daten innerhalb einer festgelegten Periode ist es auch möglich, die Firma betriebswirtschaftlich zu optimieren. Eine selbsterklärende Bedienerführung soll auch dem ungeübten Anwender eine schnelle Einarbeitung ermöglichen. Das Programmpaket Busy läuft auf IBM-kompatiblen PCs mit einem Hauptspeicher ab 640 KByte und 2 MByte freiem Speicherplatz auf der Festplatte unter MS-DOS ab der Version 2.11. Der Preis liegt zwischen 4500 und 8000 DM.

MP

Programmieren mit FORTRAN 77

von G. Bärwolff u. R. Kosciolowicz, Reihe Technische Informatik, Verlag Technik, Berlin 1990, 240 S., 6 Bilder, 17 Tafeln, Broschur, Bestell-Nr. 554 45 89

Noch immer ist die Programmiersprache Fortran aktuell, wenngleich sie bereits in den fünfziger Jahren entstanden ist. Das Buch beschreibt Fortran 77 und damit den derzeit festgeschriebenen Standard der Sprache. Auf die Weiterentwicklung zu Fortran 8X wird hingewiesen.
In der üblichen Reihenfolge werden die Sprachelemente abgehandelt. Datentypen, Variablen, Konstanten, Ausdrücke und Wertzuweisungen, Programmverzweigungen und Zyklen bilden jeweils den Gegenstand einzelner Kapitel. Die weiteren Teile sind den Zeichenketten, der Unterprogrammtechnik sowie der Ein-/Ausgabeprogrammierung gewidmet. Ein Kapitel verdeutlicht Unterschiede zu Fortran IV und zu Pascal. Im letzten Kapitel werden auch softwaretechnologische Aspekte angesprochen. Es wird ein größeres Programmbeispiel zur mehrdimensionalen Ausgleichsrechnung schrittweise entworfen und dann in allen Details ausprogrammiert. Die Teile werden einzeln übersetzt und mittels Programmverbinderlauf zusammengefügt. Damit kann zugleich die Instrumentierung eines Fortran-Programmiersystems vorgeführt werden. Die Autoren sprechen von Metakommandos.
Allen Kapiteln sind Übungsaufgaben zugeordnet. Lösungen ausgewählter Aufgaben werden im Anhang mitgeteilt. Die Autoren weisen darauf hin, daß alle Beispielprogramme unmittelbar aus einer Testumgebung in das Buchmanuskript kopiert wurden. Damit ist ein hoher Grad an Sicherheit gegen Fehler erreicht worden. Ebenfalls in der Form von Anhängen wird eine Syntaxübersicht und eine Kurzbeschreibung der Fortran 77 – Standardfunktionen mitgeliefert. Auch eine Seite mit dem ASCII-Zeichensatz fehlt nicht.
Die Autoren führen den Leser in einem gefälligen Stil an die Programmierung in Fortran 77 heran. Die Programmbeispiele sind ansprechend ohne künstliche Schnörkel ausgewählt, aber jeweils doch gut geeignet, die gerade erörterten Sprachkonstrukte zu beleuchten. *C. Polze*

Turbo-Pascal

von G. Paulin, 2., stark bearbeitete Auflage, Verlag Technik, Berlin 1990, 240 S., 28 Bilder, 29,80 DM

Turbo-Pascal ist im PC-Bereich eine der beliebtesten Programmiersprachen. Dieses Buch aus der Reihe „Technische Informatik" ist in der 2. Auflage sehr stark überarbeitet worden, um die neueren Entwicklungen von Turbo-Pascal zu berücksichtigen. Die Leistungsfähigkeit der Sprache wurde wesentlich verbessert, und die Anforderungen an die PC-Technik sind gestiegen. War die Pascal-Version 3 auch noch auf 8-Bit-PCs unter CP/M verwendbar, so setzen die neueren Versionen einen 16-Bit-PC mit MS-DOS voraus. Der Autor behandelt die Version 3 als Sprachkern, um die wesentlichen Elemente der Sprache zu beschreiben. Dieses Kapitel macht mit 150 Seiten den umfangreichsten Teil aus und entspricht im wesentlichen der 1. Auflage. Zu den eingestreuten Übungsaufgaben sind im Kapitel 5 Lösungsprogramme zu finden. Im 2. Kapitel sind die Besonderheiten der Benutzung von Turbo-Pascal 3 unter CP/M zusammengefaßt. Es werden alle Menüpunkte, die Programmentwicklung und die maschinennahe Programmierung durch Einbindung von Assemblerroutinen erläutert. Das 3. Kapitel ist neu in der 2. Auflage und behandelt Erweiterungen von Turbo-Pascal in der Version 4 (teilweise auch Version 5) unter MS-DOS. Es ist besonders für Umsteiger interessant, die vorher mit der Version 3 gearbeitet haben. Auf die Möglichkeiten des in der Version 5 integrierten Quelltext-Debuggers wird nicht eingegangen. Kapitel 4 enthält einige Informationen zur Bildschirmprogrammierung mit Version 3. Hier fehlt eine kurze Zusammenfassung der leistungsfähigen Unit GRAPH der Version 4, die das BGI (Borland Graphic Interface) realisiert. Im Anhang sind eine ASCII-Tabelle, eine Pascal-Syntax in Backus-Naur-Form, Standardfunktionen und -prozeduren, Fehlermeldungen des Compilers und des Laufzeitsystems, eine kurze Gegenüberstellung von Standard-Pascal und Turbo-Pascal, eine Aufzählung der wesentlichen Unterschiede zwischen den Versionen von Turbo-Pascal und die Editorkommandos zusammengefaßt. Literaturverzeichnis und ein übersichtliches Sachwörterverzeichnis sind am Schluß des Buches zu finden.
Besonders der Anfänger wird von den ausführlichen Beispielen und Übungsaufgaben profitieren. Als Nachschlagewerk für den versierten Programmierer ist die Darstellung weniger geeignet, da dann immer die gerade aktuelle Version interessant ist. So ist z. B. die Version 5.5 mit neuen Elementen schon wieder für eine Nachauflage zu berücksichtigen. *Hübener*

MS Makroassembler
Einführung und Leitfaden

von P. Monadjemi, Addison-Wesley (Deutschland) GmbH, Bonn 1989, 427 S., ISBN 3-89319-135-6

Gar keine Frage: Zum Erlernen der Assemblerprogrammierung benötigt man eine gute Einführung, mit den Handbüchern ist es nicht getan. Zumal, wenn es sich um einen Intel-Prozessor handelt, der mit seiner komplizierten Adreßrechnung im Segment: Offset-Format auch erfahrenen Programmierern Rätsel aufgibt – und, wenn es sich um den Microsoft-Assembler mit seiner stellenweise nicht eindeutigen Notation handelt, die Peter Monadjemi zwar in seinem Buch zum Turbo-Assembler behandelt, an dieser Stelle aber verschweigt, wohl, um dem Leser nicht den Mut zu nehmen. Der Schwierigkeiten ist er sich aber bewußt, denn sowohl der Adreßrechnung als auch der Segmentierung ist ausreichend Platz eingeräumt.
Das Buch soll in erster Linie mit den gehobenen Möglichkeiten eines modernen Makroassemblers und der zugehörigen Programme vertraut machen und nicht die Maschinenbefehle des 8086 behandeln, weshalb auf die Beschreibung der Prozessorarchitektur und eine Befehlsreferenz verzichtet wurde, die man in Handbüchern findet. Statt dessen stehen die Leistungen von MASM, LINK und CodeView im Vordergrund. Dabei sind nicht nur Einsteiger angesprochen. Auch erfahrene Assembler-Programmierer verwenden viele Direktiven nur „auf Verdacht" oder orientieren sich an Vorbildprogrammen. Fragen Sie doch einmal, wann die Distanzangaben SHORT, NEAR und FAR bei Sprüngen wirklich notwendig sind. Oder lesen Sie lieber gleich nach (S. 104). Wenn der Autor hier im Vorwort eine Lücke in den Handbüchern nennt, so muß ihm Recht gegeben werden. Dort wird zum Beispiel die Entstehung von Phasenfehlern in einem Nebensatz abgehandelt und die Möglichkeit der Nutzung von vereinfachten Segmentanweisungen für COM-Dateien ganz geleugnet. Insofern zeichnet sich das Buch nicht nur durch verständliche Darstellung, sondern auch durch Sachkenntnis aus, die nicht auf das Handbuch beschränkt bleibt. Es wird nichts ausgespart, was aus Ihnen einen guten Assembler-Programmierer macht: Prozeduren und Datenstrukturen, Macros, bedingte Assemblierung und modulares Design. Besonders hervorzuheben ist das Kapitel zur Hochspracheneinbindung. Die zahlreichen Beispiele sind bewußt kurz gehalten, um den Blick auf das Wesentliche zu richten (was auch späteres Nachschlagen unterstützt).
Aber auch die Kleinigkeiten stimmen: So hat sich der Autor Gedanken über die Verwendung von Anglismen gemacht (die er auch dem Leser nicht vorenthält). Sein Vorsatz, deutsche Bezeichnungen zu verwenden, wo immer es möglich ist, aber bei den englischen Termini zu bleiben, wo die deutsche Sprache vergewaltigt wird, sollte Nachahmung finden.
Der verbreiteten Meinung Rechnung tragend, Assemblerprogrammierer seien konservativ und Komfort gegenüber nicht aufgeschlossen, wird neben dem neuesten Stand von MASM 5.1 auch auf die – einen gewissen Standard darstellenden – Versionen 4.0 und 5.0 Bezug genommen. Den Vereinfachungen im Dschungel der Segmentdirektiven (ab 5.0) und der neuen Hochsprachenunterstützung (ab 5.1) wird man sich aber auf die Dauer nicht entziehen können. Ebenso nicht den hochsprachenähnlichen Features: Es gibt einfache String-Operationen, Textmakros sind ähnlich des C-Präprozessors einsetzbar, mit Hilfe von Pointervariablen kann der Assembler sogar die Verwendung von Zeigern überwachen. Der Einsatz derart leistungsfähiger Werkzeuge will natürlich gelernt sein, und dafür ist das Buch nur zu empfehlen. *U. Schulze*

Turbo Pascal 5.5
Kennenlernen und Beherrschen

von Christoph Kasimir, Addison Wesley, Bonn 1990, 417 S., 1 Diskette, DM 58,–, ISBN 3-89319-264-6

Immer komplexer werdende Softwarewerkzeuge sind heute mehr denn je dazu angetan, dem Einsteiger durch eine scheinbar unüberschaubare Fülle an Möglichkeiten gleich am Anfang jeden Enthusiasmus zu nehmen. Es mutet als eine unlösbare Aufgabe an, den Anfänger hier in didaktisch kluger Weise in die Problematik einzuführen und dabei die Komplexität eines Softwarepaketes vom Schlage eines Turbo Pascal 5.5 ebensowenig aus den Augen zu verlieren wie die zum Verständnis notwendige Einfachheit und Geradlinigkeit in der Darstellung. Daß diese Aufgabe lösbar ist, zeigt Christoph Kasimir mit seinem Buch, von dem man sofort angetan ist. Denn es überzeugt mit einer Sachlichkeit, die die besondere Einführungsgabe des Autors gegenüber dem Leser deutlich werden läßt. Man wird nicht mit den sonst in Einsteigerliteratur üblichen gemeinplatzartigen Floskeln strapaziert. Das Buch reiht sich nahtlos in die Tradition guter amerikanischer Fachbücher ein, denen sich der Addision Wesley Verlag in besonderer Weise verpflichtet fühlt. Wenn die wirklich guten Handbücher zu Turbo Pascal 5.5 die Information bringen, so ergänzt dieses Buch in didaktischer Hinsicht, wobei es die gesamte Palette der Turbo-Pascal-Programmierung aufzeigt und den Anfänger auf geschickte Weise an nahezu alle Probleme heranführt. Die ersten Kapitel sind betont einfach gehalten. Die Bedienung der Entwicklungsumgebung und die Erläuterung der Pascal-Sprachelemente sind im Zusammenhang dargestellt, so daß alle Beispiele sofort nachvollziehbar bleiben. Besonders hervorzuheben ist ein 45 Seiten umfassender Abschnitt zur objektorientierten Programmierung. Mit zwei weiteren speziellen Kapiteln, die die Grafikprogrammierung ebenso anschneiden wie die Erstellung von UNITs, wird eine Vollständigkeit erreicht, die den besonderen Reiz dieses Buches ausmacht.
Alle Programme sind auf einer begleitenden Diskette enthalten, wobei einige Routinen in UNITs zusammengefaßt wurden. Nicht alle der im Buch mitgeteilten Programmkonstrukte befinden sich auch so auf dem Datenträger. Einige Änderungen wurden hier doch vorgenommen. Daß es dabei nicht ganz fehlerfrei zuging, beweist die Tatsache, daß beispielsweise im Programm CRTTEST.PAS die Funktion *StringOF* aufgerufen wird, die in UNIT SYSHLP enthalten sein soll, dort aber leider nicht zu finden ist. Dies dürfe der Sache selbst aber keinen Abbruch tun, denn wer bis zu diesem Programm die Ausführungen des Autors verfolgte, wird kaum Schwierigkeiten haben, diese Routine selbst zu ergänzen. Was ihm sicher ein noch größeres Erfolgserlebnis beschert und die Richtigkeit der Kaufentscheidung für dieses Buch bestätigt.

Ch. Hanisch

Cache-Speicherkonzept für den i486

Der 32-Bit-Mikroprozessor 80486 von Intel – auch als i486 bezeichnet – unterscheidet sich von seinem Vorgänger, dem 80386, in der Hauptsache dadurch, daß er auf einem Chip den Prozessor 80386, den mathematischen Koprozessor 80387, einen 8 KByte großen Cache-Speicher und den dazugehörigen Cache-Controller vereinigt (vgl. MP 9/89, S. 278 und MP 4/90). Trotzdem gibt es Bestrebungen, zusätzlich sowohl einen Koprozessor als auch einen externen Cache anzubieten, obwohl die Onchip-Baugruppen den Vorteil der kürzeren Leitungswege haben. So hat die Firma Weitek den Koprozessor Abacus 4167 entwickelt, der bei rechenintensiven wissenschaftlichen und technischen Anwendungsprogrammen eine zwei- bis dreifache Geschwindigkeitserhöhung bringen soll.
Eine Cache-Erweiterung für den i486 liegt jetzt mit dem Steckmodul **Turbocache 486** von Intel vor, der eine weitere Leistungssteigerung von bis zu 15 % liefert. Dieser *Second-Level-Modul* ergänzt den 8-KByte-On-chip-Cache des i486. Er enthält den Cache-Controller 82485, der auf dem Modul je nach Version 64 oder 128 KByte SRAM-Cache-Speicher verwaltet (der Cache-Controller 82385 für den 80386 kann dagegen nur 32 KByte verwalten). In einem System lassen sich bis zu vier Turbocache 486 zu insgesamt 512 KByte kaskadieren – und das ohne Änderungen der Software.
Der Turbocache 486 paßt auf einen Standardstecksockel und erleichtert es dem Computerhersteller, künftige Leistungssteigerungen bereits beim Systementwurf zu berücksichtigen. Anbieter und Anwender können aber auch entsprechend den Leistungsanforderungen der unterschiedlichen Anwendungen entscheiden, ob sie den Cache-Modul einsetzen wollen. Der Turbocache 486 wird mit den Taktfrequenzen des i486 – 25 und 33 MHz – angeboten. Sein Cache-Controller enthält 500000 Transistoren, die in derselben 1-μm-CMOS-Technologie wie die der Prozessoren 80386 und 80486 hergestellt werden. Die beiden Turbocache 486-Versionen sollen 300 bzw. 400 Dollar kosten (für Tausenderstückzahlen). *MP*

Leiterplattendesign mit 32-Bit-Übertragung

PADS-2000 der CAD Software, USA, für Deutschland im Mai in München vorgestellt, ist ein leistungsfähiges PC-gestütztes CAD-System mit voller 32-Bit-Unterstützung für komplexes Leiterplattendesign und die Generierung der Entwicklungs- und Herstellungsdokumentation. Es ist auf 386er und 486er PCs lauffähig (mindestens 2 MByte RAM und VGA-Auflösung) und kann die Speichergrenzen des MS-DOS-Betriebssystems durch seinen virtuellen Speicher umgehen. Die Design-Genauigkeit liegt mit 0,001 mm wesentlich höher als bei herkömmlichen 16-Bit-Systemen. Es können Leiterplatten mit über 2000 ICs bearbeitet werden.
Erstmals ist mit dem PADS-2000-System die Integration eines auto-interaktiven Routers realisiert, der nach dem Push'n-Shove-Prinzip arbeitet. Bisher war das erst bei einer Rechnerbasis ab VAX und leistungsstärker möglich.
PADS-2000 ist kompatibel mit der neuen PADS-Logic CAE-Software und mit PADS-PCB, das mit über 9000 Installationen das weltweit erfolgreichste Leiterplatten-Design-System sein soll. *MP*

Symbolics FrameThrower

Symbolics, einer der führenden Hersteller von Workstations für die Computergrafik, forciert mit dem neuen FrameThrower die Anwendung und Verbreitung der High Definition Television (HDTV). FrameThrower und die Symbolics Workstation XL400/XL1200 bieten nach Meinung von Symbolics erstmalig die Möglichkeit, Computergrafik und HDTV-Videotechnologie miteinander zu verknüpfen. Damit erfährt der Bereich der Computer-Animation einen Qualitätssprung.
Der FrameThrower besitzt drei Vorteile: Er bietet uneingeschränkte Anwendungsmöglichkeiten im Bereich der Computersimulation und -animation; er kann uneingeschränkt die Visualisierung von Prozeßabläufen in Echtzeit ausgeben; und er basiert auf dem VMEbus, kann also an alle VMEbus-Workstations gekoppelt werden.
Der FrameThrower ist eine ganze Produktfamilie in HDTV-Qualität, die von Symbolics ständig erweitert werden wird. Das erste, jetzt verfügbare, Produkt ist der FT 100. *MP*

Kompakt-Laser für blaues Licht

Eine leistungsfähige und kompakte Quelle für blaues Laserlicht haben Wissenschaftler am Almaden-Forschungszentrum von IBM in San Jose, Kalifornien, entwickelt. Die optischen Eigenschaften sowie die Ausgangsleistungen des neuen Lasers lassen ihn für ein breites Anwendungsspektrum attraktiv erscheinen. So könnte er beispielsweise zu optischen Speichern mit der vierfachen Speicherdichte heute bekannter Systeme führen.
Zu den wesentlichen Eigenschaften des neuen, ausschließlich aus festen Materialien hergestellten Lasers zählen ein Wirkungsgrad von zehn Prozent – fünfmal mehr als bei früheren Anordnungen – sowie eine stabile, nahezu rauschfreie Ausgangsstrahlung, die zu scharf begrenzten Punkten fokussiert werden kann. Die letztgenannte Eigenschaft ist beispielsweise für optische Plattenspeicher wichtig. Sie speichern die Information als ein kodiertes Muster von Markierungen. Heutige Einheiten arbeiten mit Infrarot-Diodenlasern und schreiben damit Markierungspunkte mit einem Durchmesser von 0,8 Mikrometern. Wegen seiner kürzeren Wellenlänge kann blaues Licht zu kleineren Punkten mit 0,4 Mikrometern Durchmesser fokussiert werden und ermöglicht damit die vierfache Speicherdichte im Vergleich zu heutigen optischen Speichern.
Bis heute ist es noch nicht gelungen, einen kompakten und leistungsfähigen Laser, der direkt blaues Licht erzeugt, herzustellen. Die IBM-Wissenschaftler nutzen deshalb die Methode der *Frequenzverdopplung*, um Infrarotlicht in blaues Licht umzuwandeln. Das infrarote Licht eines Diodenlasers wird dazu durch einen winzigen, rund sechs Millimeter langen Kristall aus Kaliumniobat geschickt.
Der neue Kompaktlaser für blaues Licht wurde anläßlich der Conference on Lasers and Electro-Optics in Anaheim, Kalifornien, vorgestellt. Auf der gleichen Veranstaltung berichteten Wissenschaftler vom Almaden-Forschungszentrum über die Arbeiten an einem neuartigen Laser, der durch *Infrarot-Pumpen* und *Upconversion* blaues Laserlicht direkt – also nicht mit der Methode der Frequenzverdopplung – erzeugt.
Der *Upconversion*-Laser verwendet einen Kristall aus Yttrium-Lithiumfluorid, dem ein Prozent Erbium-Ionen (ein Element aus der Gruppe der seltenen Erden) als Dotierung hinzugefügt wurde. Er erfordert eine sehr niedrige Umgebungstemperatur (unter 20 Kelvin). Die künftige Forschung zielt jedoch auf die Entwicklung von ähnlichen Lasern bei höheren, leichter zu beherrschenden Temperaturen. Die IBM-Wissenschaftler halten es derzeit noch für verfrüht, spezielle Anwendungen für den Upconversion-Laser zu nennen. Dennoch zeigt er Möglichkeiten für neue Laser auf, die das schwer in den Griff zu bekommende, hochfrequente blaue und ultraviolette Licht erzeugen. *MP*

Dreidimensionale DRAM-Zelle

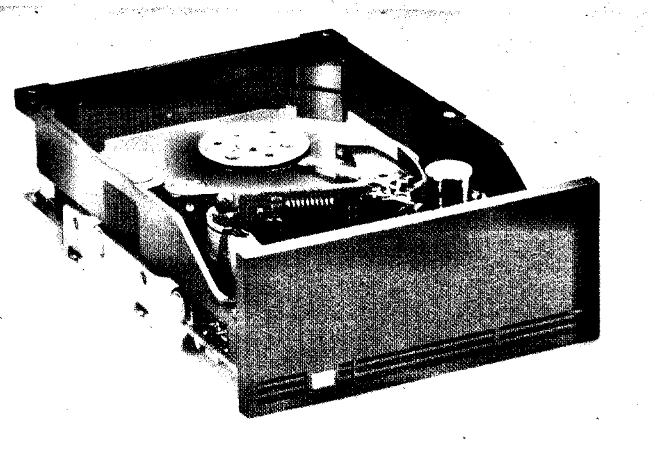
Die Entwicklung einer dreidimensionalen Speicherzelle gab die Firma Toshiba bekannt. Dabei wurden Ansteuertransistor und Speicherkondensator übereinander gestapelt. Der Prototyp wurde mit einer konventionellen 0,8-μm-Struktur entworfen. Die Firma arbeitet nun daran, die Strukturbreite auf 0,25 μm zu senken, da sich diese Zellenkonstruktion, nach Ansicht des Unternehmens, sehr gut für 1-GBit-DRAMs eignen würde. *Fa*

Supraleiterstrom von 1 Mio Ampere

An der Stanford-Universität in Kalifornien ist es gelungen, einen sogenannten kritischen Strom von 1 Mio Ampere in Hochtemperatur-Supraleitern zu erreichen. Unter dem kritischen Strom wird die Stromstärke verstanden, bei der der supraleitende Werkstoff gerade noch in seinem widerstandslosen Zustand verbleibt.
Dabei wurde festgestellt, daß bei 78 Kelvin in den vier Mikrometer dünnen Yttrium-Barium-Kupferoxiden ein Auswandern der supraleitenden Ströme begann. Durch eine Verringerung der Temperatur mit flüssigem Stickstoff auf 77 Kelvin wurde wieder ein ungestörter Stromfluß erreicht. *Fa*

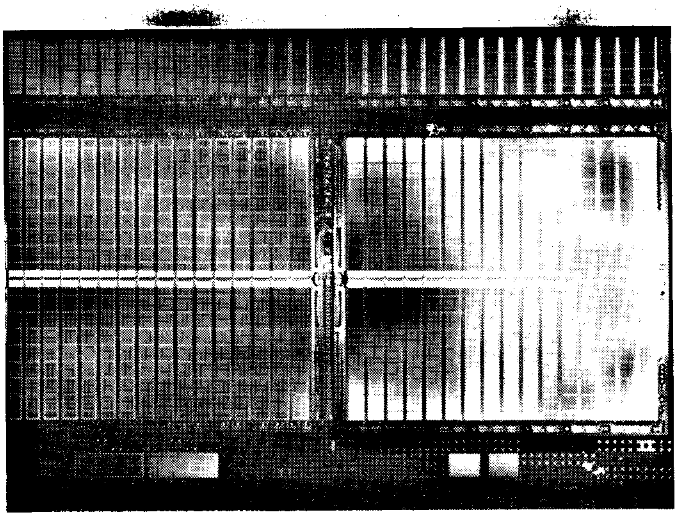
Harddisk mit neuartiger Systemkonzeption

Erstmals zur CeBIT '90 präsentierte Kyocera das im Bild dargestellte Festplattenlaufwerk **KC-40 GA** mit einer Speicherkapazität von 40 MByte, das sich durch eine neuartige Systemkonzeption auszeichnet. In diesem Laufwerk greift jeweils ein Schreib-/Lesekopf auf eine Seite des Laufwerks zu, das insgesamt nur *eine* Magnetplatte enthält. Ein Cache-Speicher in der Größe von **32 KByte** ermöglicht einen schnelleren Zugriff auf die einzelnen Daten; die mittlere Zugriffszeit beträgt 28 ms. Das Modell in der kompakten Bauform im 3,5-Zoll-Format und der Ausführung in halber Bauhöhe verfügt über eine sichere automatische Parkfunktion für die beiden Schreib-/Leseköpfe.
Das neue Laufwerk KC-40 GA verfügt über ein PC/AT-Interface (IDE embedded controller) und einen Interleave-Faktor von 1:1 für den Einbau in entsprechende PC-Systeme und ergänzt damit die Familie der zuverlässigen und leistungsstarken (20- und 30-MByte-) Festplattenlaufwerke von Kyocera.



16-MBit-DRAMs vor der Serienreife

Das Jahr 1989 war das Jahr der Produktionseinführung des 4-MBit-DRAMs. In diesem Jahr wurden von den führenden japanischen und US-amerikanischen Chip-Schmieden Prototypen des 16-MBit-DRAMs fertiggestellt. Im Rennen ist auch noch die Siemens AG, die kürzlich die Pilotfertigung des HYB 511610 aufnahm. Aber Siemens wird sicher nicht der einzige europäische Produzent von 16-MBit-DRAMs sein. Neben IBM, die bereits den 4-MBit-DRAM in der Bundesrepublik fertigt, baut Texas Instruments eine DRAM-Fabrik in Avezzano (Italien) auf, wo 1991 die Produktion der 16-MBit-Speicher anlaufen soll.
Die Chipfläche des 16-MBit-DRAMs von Siemens ist mit 142 mm^2 wesentlich größer als die seiner Vorgänger (1 MBit: 54 mm^2, 4 MBit: 91 mm^2). Er enthält aber auch die stolze Zahl von über 33 Mio Bauelementen (siehe Foto). Hergestellt wird er in einer 0,6-μm-CMOS-Technologie und ermöglicht Zugriffszeiten von 60 bzw. 80 ns; seine Verlustleistung beträgt 350 mW im Betrieb und 5 mW in Ruhe. Grundlage dieses nach zweijähriger Entwicklungszeit erreichten Erfolges ist die mit dem MEGA-Projekt in München-Perlach geschaffene Technologiebasis.
Aber Hitachi setzte kürzlich noch eins drauf: Erste Prototypen ihres 64-MBit-DRAMs sollen fertiggestellt worden sein. Seine 140 Millionen Bauelemente sind auf einer Fläche von knapp 198 mm^2 untergebracht. Die Herstellungstechnologie ist besonders bemerkenswert: Mit einem Elektronenstrahl wird eine Strukturbreite von nur 0,3 μm erzeugt. Bei der bisher üblichen Fotolithografie, die Licht verwendet, existiert eine Untergrenze von 0,5 μm. Deshalb wird beispielsweise von IBM an der Röntgenstrahllithografie gearbeitet. *MP*



Schnelles Wachstum bei OCR und Scannern

Einer Studie des Marktforschungsunternehmens Frost & Sullivan zufolge kann der westeuropäische Markt für Scanner und Klarschriftleser (OCR) mit einem hohen Umsatzwachstum rechnen. Der Absatz, so die Prognose, könnte von 18 000 Scannern und rund 10 000 Klarschriftlesern im Jahre 1988 auf etwa 230 000 Scanner und 150 000 Klarschriftleser im Jahre 1993 anwachsen. Im gleichen Zeitraum soll das Marktvolumen in Westeuropa von 33 auf über 340 Mio US-Dollar bei Scannern sowie von 6 auf 97 Mio US-Dollar bei Klarschriftlesern ansteigen. Dieses Wachstum könnte nach Ansicht von Frost & Sullivan nur erreicht werden, wenn die Hersteller die Erkennungsgenauigkeit ihrer Produkte verbessern. Die gegenwärtig erreichte Erkennungsgenauigkeit von 97 und teilweise 98 % bietet kein perfektes und zuverlässiges Leseinstrument. Bei OCR-Systemen wird eine Lesegenauigkeit von einem Fehler je 10 000 Zeichen erwartet.
Das Marktforschungsunternehmen geht davon aus, daß der OCR-Markt in der Entwicklung am weitesten fortgeschritten ist. Dabei gehören zu den aussichtsreichsten Verfahren, die zur Erhöhung der Lesegenauigkeit beitragen könnten, die Verwendung von Neuronal-Netzwerk- und von Parallel-Computern mit problemspezifischen Strukturen. Damit werde es auch möglich sein, Handschriften zu erkennen. *Fa*

Südkorea drängt auf den USA-Markt

Nach Ansicht des Vorsitzenden der amerikanischen Firma Micron Technology will Südkorea den USA-Markt mit billigen DRAMs überfluten. Diese Vermutung begründet er damit, daß Südkorea Wafer-Produktionen aufbaut, die zehnmal größer sind als es für den eigenen Markt notwendig ist. Die koreanischen Halbleiterfirmen wollen seiner Meinung nach mit Dumping-Methoden die amerikanischen Hersteller vom Markt verdrängen. Die US-Regierung wird aufgefordert, Maßnahmen dagegen zu ergreifen und vor allem von weiteren Investitionen in größere koreanische Unternehmen abzusehen. Er tritt für eine Isolation der amerikanischen Elektronikindustrie gegenüber asiatischen Ländern ein. Taiwan würde beispielsweise in etwa fünf Jahren die gleichen Ziele verfolgen wie gegenwärtig Südkorea.
Der Vorsitzende von Micron Technology sprach sich gegen das amerikanisch-japanische Handelsabkommen von 1986 aus, bei dem ein Mindestpreis für japanische DRAMs in den USA festgesetzt wurde und gleichzeitig amerikanischen Firmen eine 20prozentige Beteiligung an japanischen Firmen bis 1991 als Ziel vorgegeben wurde. Seiner Ansicht nach sollte die Beteiligung der amerikanischen Firmen bei 50 % liegen, um einem Untergang der amerikanischen Halbleiterindustrie vorzubeugen. *Fa*

Analoger Neuro-Prozessor

Die japanische Firma Matsushita Electronics entwickelte einen Neuro-Prozessor mit 64 Knoten pro Chip. Diese 64 Knoten sind in drei Ebenen angeordnet und in einer Matrix-Konfiguration verdrahtet. Der Neuro-Prozessor in BICMOS-Technologie führt 100 Mio Multiplikationen pro Sekunde aus und ist damit 500- bis 1000mal schneller als konventionelle Signalprozessoren. Die Firma will in den nächsten zwei bis drei Jahren die Integrationsdichte bei den Neuroprozessoren auf 256 Knoten pro Chip erhöhen. Außerdem soll ein System für den Einsatz in Zeichenerkennungsapplikationen entwickelt werden, das aus drei Chips besteht. *Fa*

Ultranet mit 1 GBit/s

Eine neue Netz-Architektur, mit der es möglich sein soll, Übertragungsgeschwindigkeiten von 1 GBit pro Sekunde zu erreichen, wurde von der kalifornischen Firma Ultra Network Technologies Inc. entwickelt. Dieses Rechnernetz – mit der Bezeichnung Ultranet – wurde mit RISC-Rechnern der Firma MIPS aufgebaut. Es soll vorwiegend bei RISC-Prozessoren mit höheren Netzgeschwindigkeiten eingesetzt werden.
Die Firma MIPS will demnächst RISC-Rechner auf den Markt bringen, die mit 20 MIPS arbeiten, und im Anschluß daran werden RISC-Rechner mit 55 MIPS entwickelt. Die Netzsoftware von Ultranet soll zu einem Preis von 8500 US-Dollar erhältlich sein. *Fa*

Token-Ring-Controller auf EISA-Basis

Die Compaq Computer Corporation stellte im April den *Compaq 32-Bit DualSpeed Token Ring Controller* vor. Dieser Controller ist derzeit der leistungsstärkste Netzwerk-Schnittstellen-Controller für Server in Token-Ring-Netzwerken. Beim Einsatz im Compaq Systempro oder Compaq Deskpro 486/25 kann der 32-Bit-DualSpeed-Token-Ring-Controller die Leistungsvorteile von EISA (Erweiterte Industrie-Standard-Architektur) voll nutzen. Er ermöglicht so um bis zu 32 % höher liegende Übertragungsgeschwindigkeiten. Diese extrem hohe Leistung ist insbesondere in anspruchsvollen Server-Umgebungen wichtig, zum Beispiel wenn der Server als Brücke zwischen mehreren Sub-Netzwerken dient oder im Client/Server-Einsatz.
Der 32-Bit-DualSpeed-Token-Ring-Controller setzt die 32-Bit-EISA-Busmaster-Technik ein und wickelt den Datentransfer vom und zum Arbeitsspeicher in sogenannten Burst-Raten von bis zu 33 MByte pro Sekunde ab. Dies optimiert die Prozessorleistung, so daß der Prozessor länger für die Aufgaben von Anwendungsprogrammen zur Verfügung steht.
Zur weiteren Verbesserung der Netzwerk-Performance können Anwender bis zu fünf dieser Token-Ring-Controller im Systempro oder im Deskpro 486/25 installieren und so die Leistung des Systems exakt an ihre Anforderungen anpassen. So bringt beispielsweise ein zweiter Controller im Compaq Systempro eine Erhöhung des Datendurchsatzes um 99 %.
Der 32-Bit-DualSpeed-Token-Ring-Controller verfügt über optimierte Treiber für eine Reihe von weitverbreiteten Netzwerk-Betriebssystemen wie Novell NetWare 386, Novell NetWare 2.1x oder den LAN Manager. *MP*

Toshiba mit 4-MBit-SRAM

Der größte Speicherchip-Hersteller – die japanische Toshiba Corporation – konnte kürzlich die Entwicklung eines 23 Nanosekunden schnellen *statischen* 4-MBit-RAMs in einer 0,5-μm-CMOS-Technologie melden. Statische RAMs (SRAMs) haben gegenüber den DRAMs den Vorteil, daß ihr Inhalt nicht zyklisch aufgefrischt werden braucht und deshalb bei ausgeschaltetem Gerät mit Akkus gepuffert werden kann. Wegen des größeren Flächenbedarfs erreichen sie aber nicht die Integrationsdichte der DRAMs. Die für Toshibas neuen SRAM entwickelten Dünnfilmtransistoren erreichen gegenüber ihren Vorgängern eine geringere Leistungsaufnahme und eine höhere Zuverlässigkeit. Der Ruhestromverbrauch des Chips im Pufferbetrieb beträgt 1 μA. Trotzdem hat der Schaltkreis die stattliche Arbeitsstromaufnahme von 20 mA bei einer Taktfrequenz von 10 MHz und von 70 mA bei 40 MHz. Die 25 Millionen Transistoren, von denen je 6 eine 20 μm^2 große Speicherzelle bilden, befinden sich auf einem Chip mit einer Fläche von rund 136 mm^2; die Organisation der über 4 Millionen Speicherzellen ist 512 K × 8 Bit. *MP*

Virtuelles LED-Display

Die amerikanische Firma Reflection Technology Inc. stellte ein neuentwickeltes Miniatur-Display vor, welches man, bezogen auf die Funktionsweise, mit elektromagnetischen Treibern, die die Membran in Lautsprechern bewegen, vergleichen kann. Das darzustellende Bild wird über eine lineare LED-Zeile (nicht eine XY-Matrix) sowie ein Linsensystem dem Umlenkspiegel zugeführt. Dieser Umlenkspiegel „bewegt" sich vibrierend um 15 Grad hin und her. Das mit den Spiegelbewegungen zeilenweise erzeugte Bild des Displays ist virtuell. Das bedeutet, es existiert nur im Aufnahmevermögen des Betrachters.
Trotz der Nachteile, die die Spiegelmechanik mit sich bringt, gibt es eine Reihe von Vorteilen wie: große Helligkeit, in relativ weitem Rahmen frei wählbare Bildgröße und geringes Volumen. Als Anwendungsgebiete dieses Displays sieht die Firma neben papierlosen Telefaxgeräten auch Head-up-Displays für Fahr- und Flugzeuge. *Fa*

Computer-Club

Compilerkompatible Quelltexte in Turbo Pascal

Kay Hradilak, Berlin

Anwendungsprogramme müssen häufig auf verschiedenen Rechnern und unter verschiedenen Betriebssystemen lauffähig sein.
Damit entsteht das Problem – wenn man nicht dasselbe Programm mehrfach entwickeln will –, Quelltexte so zu schreiben, daß sie von verschiedenen Compilern verarbeitet werden können und als Programme in ihrer inhaltlichen Wirkung identisch sind. Dies soll zumindest für die Versionen eines Compilers (z. B. Turbo-Pascal) gelten, prinzipiell aber auch für verschiedene Compiler einer Sprache möglich sein.
Compiler arbeiten auf bestimmten Rechnern unter bestimmten Betriebssystemen. Das Problem erwächst also zuerst aus der Existenz unterschiedlicher Hardware, deren besondere Möglichkeiten programmsprachlich zu berücksichtigen sind. Für die gleiche Hardware sind in der Regel mehrere Betriebssysteme verfügbar (z. B. MS-DOS, XENIX, OS/2).
In ihrer Bezeichnung und in ihrer Bedeutung abweichende Sprachelemente sind auch durch die Folge von Versionen für gleiche Maschinen und gleiche Betriebssysteme bedingt (Turbo 3.0 –> Turbo 4.0!).
Man kann sagen:
WENN eine Programmiersprache auf einer unterschiedlichen Hardware
UND/ODER mit unterschiedlichen Betriebssystemen
UND/ODER in unterschiedlichen Compilern
UND/ODER in unterschiedlichen Versionen implementiert ist,
DANN besteht die Frage
der Sicherung der syntaktischen und semantischen Übereinstimmung von Quelltexten, kurz ihrer Compilerkompatibilität bzw. Portabilität.
Eine befriedigende Antwort kann die Anwendung zweier Prinzipien geben, die ich als Low Level Moduling und Obermengenorientierung bezeichnen möchte.
Low Level Moduling ist die Realisierung bestimmter maschinen- und systemspezifischer Operationen (Begriffsinhalt) in Unterprogrammen mit allgemeiner für alle Compiler geltender Nutzeroberfläche (Begriffsbezeichnung), die in Modulen zusammengefaßt sind. Es wird in Quelltexten, die durch die Bezeichner auf diese Operationen verweisen, solch eine syntaktische Übereinstimmung ermöglicht, daß sie entsprechend des unterschiedlichen Kontextes (Hardware, Betriebssystem, Compiler) eine gleiche Bedeutung (semantische Übereinstimmung) haben. Dies muß auch für allgemeine Typen, Konstanten und Variable gelten.
Low Level Moduling wird bereits durch die Standardbibliothek der jeweiligen Compiler verwirklicht:
So hat die Verwendung des Bezeichners **CLRSCR** in allen Turbo-Pascal-Versionen ungeachtet zugrundeliegender Hardware und Betriebssysteme die gleiche Wirkung – das Löschen des Bildschirmes.
Obermengenorientierung heißt, maschinen- und systemspezifische Operationen sowie allgemeine Typen, Konstanten und Variablen, die in Low-level-Modulen zusammengefaßt sind, bezüglich der gültigen Nutzeroberfläche an den Erfordernissen der Obermenge des Leistungsangebotes zu orientieren. Bei Maschinen und Betriebssystemen mit geringerem Leistungsangebot (Tastatur, Dateiarbeit, optische Darstellung, Fehlerbehandlung u. a.) sind diese Operationen als Teilmenge des höchst verfügbaren Maßstabes zu betrachten und zu programmieren. Compiler verwirklichen dieses Prinzip nicht ausdrücklich. Die Anwendung dieser Prinzipien wird an der Behandlung der Tastatur beim PC 1715 mit CP/M-kompatiblen Betriebssystemen und bei PCs unter MS-DOS demonstriert. Die Prozeduren für die einzelnen Compiler können dem Listing entnommen werden.

Compilerkompatible Tastaturbehandlung
Die Tastaturbehandlung unterschei-

```
  1:
  2:      {****************************************************
  3:       *            (C)---->k.hradilak '89   19/10/89     *
  4:       *                                                  *
  5:       *              PROGRAM    KEYDEMO                  *
  6:       *              MODULE                              *
  7:       *                                                  *
  8:       ****************************************************}
  9:
 10:
 11: PROGRAM KEYDEMO ;
 12:  uses crt; { NUR TURBO >= 4.0 !!}
 13:
 14:
 15:    {" ALLGEMEINE KONSTANTEN UND TYPEN ZUR TASTATURBEHANDLUNG  "}
 16:
 17:    {  Die aufgeführten Konstanten stellen nur eine Auswahl dar.
 18:       So fehlen zum Beispiel sämtliche ALT - und CTRL - Kombi-
 19:       nation. Je nach Bedarf - Verwendeter Compiler ! - sollten
 20:       sie im Quelltext verbleiben. }
 21:
 22:   CONST
 23:         { Bewegungstasten : }
 24:         UP=#72;DOWN=#80;LEFT=#75;RIGHT=#77;HOME=#71;
 25:         EEND=#79;PGUP=#73;PGDN=#81;
 26:
 27:         INS=#82;DEL=#83;ESC=#27;CR=#13;
 28:
 29:         { Funktionstasten :}
 30:         F1=#59;F2=#60;F3=#61;F4=#62;F5=#63;
 31:         F6=#64;F7=#65;F8=#66;F9=#67;F10=#68;
 32:         { Zeichenmengenkonstanten : }
 33:         digs:SET OF char=['0'..'9'];
 34:         lets:SET OF char=['a'..'z','A'..'Z'];
 35:         ascii:SET OF char=[' '..'z'];
 36:         JN  :SET OF char=['j','J','n','N'];
 37:         noset: SET OF char=[];
 38:         { Umwandlung Klein- in Großbuchstaben ? }
 39:         upcaseon=true; upcaseoff=false;
 40:  TYPE
 41:      defsign_s =SET OF char;
 42:      usedkey   =ARRAY[1..2] OF char;
 43:  CONST
 44:      STANDSIGN = 1; SPECSIGN  = 2;
 45:
 46:
 47:    {" ZEICHENIDENTIFIKATIONSROUTINE FUR TURBO PASCAL 4.0/5.0 " }
 48:
 49:   PROCEDURE getkey(VAR key:usedkey) ;
 50:
 51:    BEGIN{ proc getkey() }
 52:     key[SPECSIGN]:=#0;
 53:     key[STANDSIGN]:=readkey;
 54:     IF key[STANDSIGN]=#0 THEN
 55:      BEGIN
 56:       key[SPECSIGN]:=readkey;
 57:      END;{ if key[STANDSIGN]=#0 }
 58:    END;{ proc getkey() }
 59:
 60:
 61:  {" ZEICHENIDENTIFIKATIONSROUTINE FUR TURBO PASCAL 3.0 - MS-DOS "
 62:
 63:  PROCEDURE getkey(VAR key:usedkey) ;
 64:   VAR
 65:      ch:char;
 66:    BEGIN{ proc getkey() }
 67:      key[SPECSIGN]:=#0;
 68:      read(kbd,key[STANDSIGN]);
 69:      IF key[STANDSIGN]=ESC THEN
 70:       IF keypressed THEN
 71:       BEGIN
 72:        key [STANDSIGN]:=#0;
 73:        read(kbd,key[SPECSIGN]);
 74:       END;{ if keypressed }
 75:     END;{ proc getkey() }
 76:
 77:
 78:  {" ZEICHENIDENTIFIKATIONSROUTINE FUR TURBO PASCAL 3.0 - CPM 2.2"}
 79:   { Genutzt auf dem PC 1715 }
 80:
 81:   PROCEDURE getkey(VAR key:usedkey) ;
 82:    BEGIN{ proc getkey() }
 83:     key[SPECSIGN]:=#0;
 84:     read(kbd,key[STANDSIGN]);
 85:     IF  key[STANDSIGN] IN [^E,^X,^H,^D,^Q,^R,^C] THEN
 86:      BEGIN
 87:      IF key[STANDSIGN]=^Q THEN
 88:       BEGIN
 89:        read(kbd,key[SPECSIGN]);
 90:        CASE key[SPECSIGN] OF
 91:         ^R:key[SPECSIGN]:=HOME;
 92:         ^C:key[SPECSIGN]:=EEND;
 93:         ELSE key[SPECSIGN]:=#0
 94:         END; {case key[SPECSIGN] }
 95:       END
 96:       ELSE
 97:        CASE key[STANDSIGN] OF
 98:         ^E:key[SPECSIGN]:=UP;
 99:         ^X:key[SPECSIGN]:=DOWN;
100:         ^H:key[SPECSIGN]:=LEFT;
101:         ^D:key[SPECSIGN]:=RIGHT;
102:         ^R:key[SPECSIGN]:=PGUP;
103:         ^C:key[SPECSIGN]:=PGDN;
104:        ELSE key[SPECSIGN]:=#0;
105:       END;{ case key[STANDSIGN] };
106:      key[STANDSIGN]:=#0;
107:     END;{if  key[STANDSIGN] in []   }
108:    END;{ proc getkey() }
109:
110:
111:    {" ALLGEMEINE COMPILERKOMPATIBLE ROUTINE ZUR ABFRAGE EINER
112:      ENTSCHEIDUNG "}
113:
114:    PROCEDURE getdec(y1,x1:byte;VAR decis:usedkey;k1,k2:defsign_s;
115:                     upcaseon:boolean) ;
116:     BEGIN{ proc getdec() }
117:      REPEAT{ -->(decis[STANDSIGN] in k1) or (decis[SPECSIGN]  }
118:       gotoxy(y1,x1);
119:       getkey(decis);
120:       IF upcaseon THEN
121:        decis[STANDSIGN]:=upcase(decis[STANDSIGN]);
122:      UNTIL (decis[STANDSIGN] IN k1) OR (decis[SPECSIGN] IN k2);
123:     END;{ proc getdec() }
124:
125:
126:              {"  ANWENDUNG IN EINER MENU - SITUATION "}
127:
128:  CONST
129:   firsty    = 30;
130:   firstx    = 12;
131:   maxchoice = 4;
132:
```

det sich bei der verglichenen Hardware und dem jeweiligen Betriebssystem erheblich, es gibt aber auch Unterschiede bei den MS-DOS-Versionen von Turbo-Pascal.
In den meisten Anwendungsprogrammen ist die Tastatur das Hauptinstrument der Nutzersteuerung.
Eine sichere, eindeutige Nutzersteuerung setzt als erstes voraus, daß in Entscheidungs- oder Erfassungssituationen vom Programm gewartet wird, bis eine Taste bzw. Tastenkombination betätigt wurde. Diese muß vom System identifiziert und an eine Variable des Anwendungsprogrammes übergeben werden. Dafür stehen in den Versionen bis 3.0 die Prozedur **read(kbd,var ch :char)** und ab 4.0 die Funktion **readkey :char** zur Verfügung.
Weiterhin dürfen in jeder Entscheidungs- oder Erfassungssituation nur die Zeichen zulässig sein, die Programmaktionen auslösen sollen (z. B. '0'..'9' zur Funktionsauswahl, Ja/Nein – Abfragen oder Buchstaben zur Texterfassung). Jede allgemeine Tastaturabfrageroutine kann also folgende Bedeutung bzw. Inhalt haben:

WIEDERHOLE SOLANGE
– Warte bis Taste oder Tastenkombination betätigt.
– Identifiziere Zeichen und übergebe es an eine Programmvariable
BIS ÜBERGEBENES ZEICHEN IN ZULÄSSIGER MENGE

Eine solche Routine muß als atomare Operation eine Zeichenidentifikationsroutine erhalten. Diese Prozedur soll **getkey()** heißen. Sie kann zum Beispiel in einer Routine zur Abfrage einer Entscheidung – **procedure getdec()** – genutzt werden.
Die Syntax bzw. die Nutzeroberfläche beider Routinen enthält folgende Parameter:
① Prozedur getkey()
– eine Variable zur Übernahme des jeweiligen Zeichenkodes
② Prozedur getdec()
– eine Variable zur Übernahme des jeweiligen an getkey() zu übergebenden Zeichenkodes
– die zulässige Zeichenmenge
und als zusätzliche Funktion:
– die Ermittlung der Bildschirmposition bei der Tastaturabfrage
– die (boolesche) Festlegung, ob eine automatische Umwandlung von Klein- in Großbuchstaben erfolgen soll.

Beide allgemeinen Routinen sind nun inhaltlich und formell bestimmt. Damit ist ein Low-level-Moduling im wesentlichen möglich.
Der höchste verfügbare Maßstab der betrachteten Versionen wird durch die MS-DOS-Compiler bereitgestellt:
Der Zeichencode wird vom System als 2-Byte-Ausdruck gespeichert. Die beiden Bytes sind dabei meist wechselseitig auf Null gesetzt. Es gilt:
Wenn das niederwertige Byte <> 0 ist, dann handelt es sich um eine Standardtaste bzw. ein Zeichen des erweiterten ASCII-Kodes, ansonsten wurde eine Sondertaste betätigt (z. B. F1).
Es kann auch der Umschaltstatus (Shift) festgestellt werden, sowie welche der doppelt vorhandenen Tasten gedrückt wurde.
Uns soll vor allem der 2-Byte-Ausdruck interessieren. Um die benutzte Taste bzw. Tastenkombination eindeutig zu bestimmen, kann ein Zeichenfeld mit 2 Elementen definiert werden:
TYPE
usedkey = array [1..2] of char;
Vereinbart wird, daß das 1. Element Zeichen des Standard- bzw. des erweiterten ASCII-Kodes und das 2. Element Zeichen des Sonderzeichensatzes speichert:
CONST
STANDSIGN = 1;
SPECSIGN = 2;
Wenn also eine Sondertaste bzw. Tastenkombination gedrückt wurde, ist das 1. Element gleich #0, im anderen Falle erhält das 2. Element den Wert #0 zugewiesen. Die beiden Routinen können nun programmsprachlich formuliert werden.
Zur Nutzung der Sonderzeichen sind allgemeine Konstanten vereinbart. Sie sind auch eine Obermenge der möglichen Sonderzeichen.
Entsprechend der gleichen Hardware und des gleichen Betriebssystems erfolgt die Zeichenidentifikation in der MS-DOS-Version von Turbo 3.0 analog zu den Versionen 4.0 und 5.0. Es gilt jedoch, die syntaktischen und semantischen Unterschiede der Standardunterprogramme **read (kbd,var ch:char)** und **readkey:char** in **getkey()** zu berücksichtigen. Beim PC 1715 mit seinen CP/M-kompatiblen Betriebssystemen steht allgemein der einfache ASCII-Kode mit seinen 127 Zeichen zur Verfügung.
Ein einfaches Zeichenfeld würde also als Übergabeparameter in getkey() und getdec() genügen; doch im Sinne der Compilerkompatibilität wollen wir eine allgemeine Nutzeroberfläche und eine Orientierung an der Obermenge des Leistungsangebotes wahren.
Das setzt aber voraus, daß nach den Erfordernissen des jeweiligen Anwendungsprogrammes die Bedeutung bestimmter Tastendrücke *umgelenkt* wird (Semantikverschiebung).
Die allgemeinen Zeichenkonstanten der MS-DOS-Versionen werden – soweit sie genutzt werden sollen – übernommen.
In dem Beispiel wird der Kode der Kursortasten entsprechend ihrer 16-Bit-Darstellung in das 2. Element umgelenkt. ^R und ^C werden als Betätigen der PgUp bzw. der PgDu-Taste betrachtet. Die Tastenfolgen ^Q^R und ^Q^C sollen die Bedeutung der Tasten Home bzw. End haben.
Eine vorgeschlagene Nutzung der Funktionstasten /6/ erwies sich als problematisch und insbesondere zwischen CP/A und SCP als nicht kompatibel.
Die atomare Operation getkey() garantiert, daß die Tastaturnutzung beim PC 1715 in einem gewünschten und möglichen Umfang an dem höchsten verfügbaren Maßstab orientiert ist. Es können nun Anwendungsprogramme geschrieben werden, deren Quelltexte in bezug auf die Tastaturbehandlung compilerkompatibel sind.

```
133:  VAR
134:   deckey :usedkey;
135:   oldmenpos,
136:   menpos,
137:   actx    :byte;
138:
139:  BEGIN{ prog KEYDEMO }
140:   { .... }
141:   clrscr;
142:   gotoxy(firsty,firstx);  write('+--------------------+');
143:   gotoxy(firsty,firstx+1);write('¦      ERFASSUNG     ¦');
144:   gotoxy(firsty,firstx+2);write('¦      LESEN         ¦');
145:   gotoxy(firsty,firstx+3);write('¦      VERÄNDERN     ¦');
146:   gotoxy(firsty,firstx+4);write('¦      DRUCKEN       ¦');
147:   gotoxy(firsty,firstx+5);write('+--------------------+');
148:   menpos:=1;oldmenpos:=1;
149:   REPEAT{ -->deckey[STANDSIGN] in [CR,ESC] }
150:    gotoxy(succ(firsty)+1,firstx+menpos);write('-->');
151:    getdec(succ(firsty),firstx+menpos,deckey,[CR,ESC],
152:          [UP,DOWN,PGUP,PGDN],upcaseoff);
153:     IF deckey[SPECSIGN]<>#0 THEN
154:      BEGIN
155:       CASE deckey[SPECSIGN] OF
156:        UP  :IF menpos>1 THEN
157:              dec(menpos)
158:             ELSE
159:              menpos:=maxchoice;
160:        DOWN:IF menpos< maxchoice THEN
161:              inc(menpos)
162:              ELSE
163:               menpos:=1;
164:        PGUP :menpos:=1;
165:        PGDN :menpos:=maxchoice;
166:        ELSE BEGIN END
167:       END;{ case deckey[SPECSIGN] }
168:      END;{ if deckey[SPECSIGN]<>#0 }
169:    gotoxy(succ(firsty)+1,firstx+oldmenpos);write('   ');
170:    gotoxy(succ(firsty)+1,firstx+menpos);   write('-->');
171:    oldmenpos:=menpos;
172:   UNTIL deckey[STANDSIGN] IN [CR,ESC];
173:   IF deckey[STANDSIGN] = ESC THEN
174:    menpos:=0;
175:   { .... }
176:  END.{ prog KEYDEMO }
```

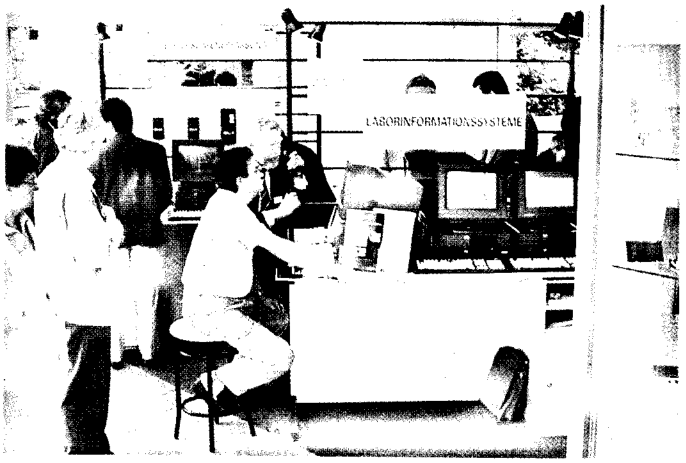
Effektive Verwaltung von Datenelementen

In Datenbankprojekten sind in der Regel mehrere Dateien zu verwalten, die teilweise gleiche Datenelemente enthalten. Ein einfaches Verfahren zur Verwaltung der Datenelemente mehrerer Dateien sowie des Dateiaufbaus für einen differezierten Informationsbedarf und zur Dokumentation wird an einem Beispiel beschrieben. Die Verwaltungsdatei ELEMENTE hat den Aufbau nach Tafel 1. Jeder Satz dieser Datei beschreibt ein Datenelement. Über die logischen Variablen *DATEI_1, ..., DATEI_...* wird markiert, in welcher der zum Projekt gehörigen Dateien dieses Datenelement benutzt wird.
Das Feld *DATUM* wird zur Registratur der letzten Satzänderung genutzt. *BEDEUTUNG* dient der verbalen Beschreibung des Datenelementes. Nach der Veränderung der Sätze in *ELEMENTE* kann das gesamte Dateiensystem leicht durchgängig neu gestaltet werden:
1. Daten der DATEI_1 auf HILF_1 (erste Hilfsdatei) kopieren (falls notwendig).
2. **use ELEMENTE**
store "DATEI_1" to M_HILF
copy to HILF_2 field FELD_NAME,;
FELD_TYP, FELD_LAEN, FELD_DEZ;
for &M HILF .and. .not. PARAMETER
3. **create &M_HILF_2**
4. Daten aus Hilfsdatei an neue DATEI_1 anhängen.
Analog ist mit DATEI_2 bis DATEI_ ... zu verfahren. Der entsprechende Zyklus ist unter Nutzung des *DATUM*-Datenelementes als Endekriterium leicht programmierbar.
Vor dem dritten Schritt kann der Name der Feldelemente in der Datei HILF_1 gegebenenfalls dateispezifisch modifiziert werden (das Datenelement *ELN* wird so zum Beispiel in DATEI_1 zu *A_ELN* und in DATEI_2 zu *B_ELN* usw.). Damit ist die Verwendung von Alias-Vorsätzen überflüssig.
In die Datei ELEMENTE wurde auch das Datenelement *PARAMETER* aufgenommen. Durch die Möglichkeit der Kommentierung und die Nutzung der Feldelemente *FELD_TYP, ...* kann dies vorteilhafter als die Verwendung einer Merk-Datei sein. Damit wird eine durch externe Parameter oder Variablen gesteuerte Programmierung und eine einfache Programmanpassung möglich:
– Ablauf- oder Variantensteuerung durch Übernahme von PARAMETER
– datenelementabhängige Operationen (z. B. Textkettenverarbeitung und Bildschirmgestaltung) durch Übernahme der Beschreibungselemente *FELD_TYP, FELD_LAEN* und *FELD_DEZ*
Die Datei ELEMENTE wird darüber hinaus für weitere Zwecke (z. B. Kennzeichnung einer Indexierung nach dem Dateiaufbau genutzt.

Dr. Jochen Kayser

Tafel 1 Verwaltungsdatei ELEMENTE

FELD_NAME	FELD_TYP	FELD_LAEN	FELD_DEZ
FELD_NAME	C	10	0
FELD_TYPE	C	1	0
FELD_LEN	N	3	0
FELD_DEC	N	3	0
BEDEUTUNG	C	20	0
PARAMETER	L	1	0
DATEI_1	L	1	0
DATEI_2	L	1	0
DATEI_3	L	1	0
DATEI_...	L	1	0
DATUM	D	8	0



COM '90

Als Messe mit einem eigenen Charakter hatten die Veranstalter der ersten deutsch-deutschen Computermesse ihr Projekt konzipiert. Ausgangspunkt war der künftige große Bedarf für den Computereinsatz, der auf dem Gebiet der DDR erwartet wird. Ziel sollte erklärtermaßen nicht das Bestreben möglichst hoher Verkaufsabschlüsse sein, sondern die Kontaktaufnahme mit potentiellen Partnern sowie die Beratung. Diesem Anliegen entsprach auch der begleitende Fachkongreß.
Als Veranstalter der vom 30. Mai bis zum 1. Juni laufenden Messe fungierten die Kammer der Technik Erfurt und die Firma Dr. Anton & Partner aus der Bundesrepublik.
Zwar waren kaum „ganz große" Hard- oder Softwareanbieter vertreten (Ausnahmen: Apple und Tektronix), aber unter den etwa 150 Ausstellern gab es doch viele bekannte Namen, angefangen von Acer, Adcomp und Altos über Goupil und Peacock bis zu RAIR und Volkswriter. Daneben boten etliche Beratungsunternehmen ihre Leistungen an. Das ausgewählte Messegelände, die iga in Erfurt, und der Termin Ende Mai mit herrlichem Wetter boten die besten Voraussetzungen zum Gelingen der Veranstaltung.
Dennoch hatten wir den Eindruck, daß die von den Veranstaltern prognostizierte Zahl von 50 000 Besuchern bei weitem nicht erreicht wurde – zumal sich die Messebesucher noch mit den Urlaubern und Schulklassen vermischten, die sich als Besucher der iga an den herrlichen Blumen ergötzten. Die noch fehlende Professionalität, die sich in einigen organisatorischen Anlaufschwierigkeiten zeigte, dürfte kaum der Grund gewesen sein.
Trotzdem läßt sich aber vielleicht beispielsweise die Aufmerksamkeit gegenüber der Fachpresse bei einer nächsten COM noch steigern.
Zu wünschen ist der Veranstaltung künftig jedenfalls ein größerer Zuspruch. *MP-We*

Hannover Messe Industrie '90

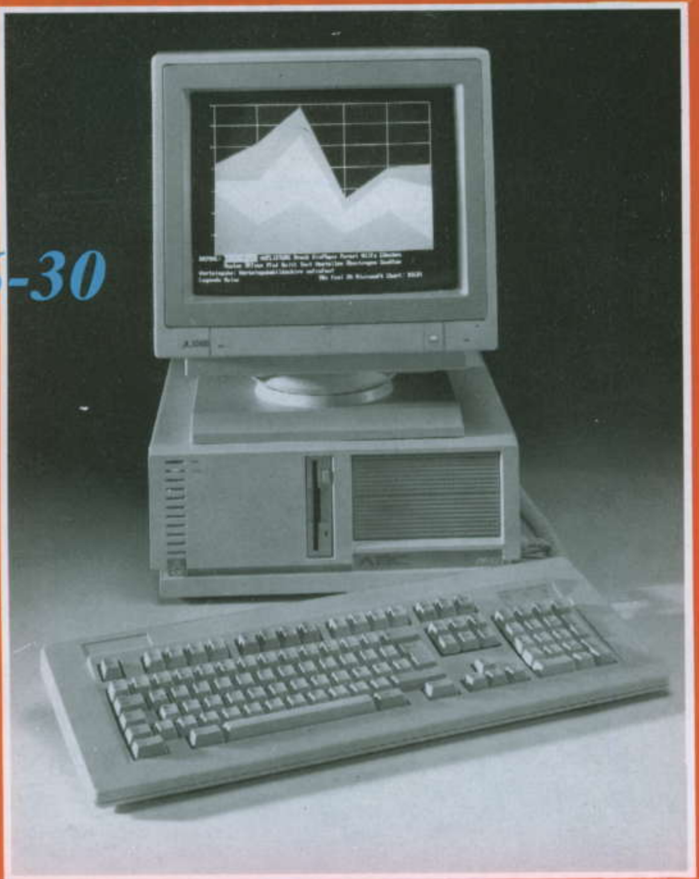
Also lag es doch nicht nur an den milden Frühlingslüften des Maimonats, daß mir der Besuch der Hannover Messe Industrie nach der wenige Wochen zuvor stattgefundenen CeBIT so viel geruhsamer, beschaulicher und damit nervenschonender vorkam: Der Abschlußbericht der Messe AG bestätigt es – gegenüber 560 000 Besuchern der CeBIT '90 drängte es „nur" 390 000 zu den Neuheiten der HMI (HMI = Hannover Messe Industrie. Was wären wir ohne Abkürzungen!), das sind auch weniger als im Vergleichsjahr 1988 mit 445 000 Besuchern. Dabei kamen immerhin über 30 000 Fachbesucher aus der DDR.
Die HMI ist derzeit in 14 Fachmessen untergliedert, die von einem umfangreichen Rahmenprogramm begleitet werden. Für Mikroelektronik- und Computerinteressenten sind vor allem die Hallen 13 und 18 von Interesse. In letzterer wurden von etwa 200 Ausstellern auf etwa 7000 m^2 die *C-Techniken für die Fertigung* präsentiert; besonders im Hinblick auf das Zusammenwirken der einzelnen C-Komponenten wie CAD, CAM, CAQ usw. Zentraler Blickpunkt der Halle war auch diesmal ein sogenanntes CIM-Leitzentrum, wo eine funktionsfähige Modellfabrik mit kundenspezifischer Fertigung aufgebaut war. An diesem Projekt, bei dem der für CIM notwendige durchgängige Informationsfluß an zwei Beispielen veranschaulicht wurde, beteiligten sich 25 Firmen. Auch wenn von offizieller Seite diese Art Präsentation als Erfolg bewertet wurde und zweifellos das Anliegen lobenswert ist, das Zusammenspiel so vieler Elemente verschiedener Hersteller beim CIM zu zeigen – es drängte sich doch die Frage nach dem Verhältnis von Aufwand zu Nutzen auf; und das Besucherinteresse hielt sich ja auch in Grenzen. (Übrigens: Ohne im geringsten etwas unterstellen zu wollen, mich erinnerte der Stand doch sehr an die Halle 15 der Leipziger Messe, in der man unseren ehemaligen politischen Größen regelmäßig die scheinbare Leistungsfähigkeit von irgend etwas „demonstrierte".)
Aber nicht nur im CIM-Zentrum, sondern auch an vielen anderen Ständen zeigten die Aussteller – beispielsweise WANG und Prime – vor allem die Anwendung von CAD. Generell kam hier also hauptsächlich der Interessent auf seine Kosten, der auf der Suche nach Lösungen – Hardware plus Software – war. Eine weitere, allerdings kleinere Präsentation verschiedener Hersteller gab es auf einem Gemeinschaftsstand, wo die Anwendung des Profibus gezeigt wurde. Als nationale Feldbusnorm sollen seine Teile 1 (OSI-Schichten 1 und 2) und 2 (OSI-Schicht 7) nun veröffentlicht werden. Da der Profibus als offener Feldbus Geräte verschiedener Hersteller vernetzt, kommt auch ihm beim CIM eine gewisse Bedeutung zu.
In der genannten Halle 13 zeigten etwa 260 Unternehmen auf der Fachmesse *MICROTRONIC – Bausteine der Elektronik* Bauelemente und Baugruppen der Elektronik, Komponenten der Optoelektronik und Sensortechnik, Mikrocomputersysteme und Softwareprodukte. Die allgemeine Leitlinie wurde von dem Einfluß bestimmt, den die Mikroelektronik heute auf praktisch alle Industriebereiche nimmt und die Chancen, die sie durch die Einbeziehung mikroelektronischer Komponenten und Systemlösungen der Industrie bietet. Auf einer speziellen VMEbus-Straße ließ sich an den vielen Ständen wiederum die Bedeutung dieses Bussystems im industriellen Bereich erkennen. Im Microtronic Transputer-Center zeigten etwa zehn Firmen – darunter Inmos, Hema und parsytec/paracom – nicht nur ihre neuesten Produkte, sondern informierten und berieten auch über die Einsatzmöglichkeiten und Leistungsfähigkeit von Transputersystemen. Von Atari war hier übrigens mit der Atari Transputer Workstation ATW eines der wenigen explizit ausgestellten Mikrocomputersysteme zu sehen. Eine größere Exposition solcher Systeme gab es nur am Stand von Kontron, wo „der erste und kleinste tragbare Industrierechner mit EISA-Bus", der IPLite, präsentiert wurde. Wahlweise kann in den 386/486-Portable sogar noch der 64-Bit-RISC-Prozessor Intel 860 eingesetzt werden, ebenso wie in den gezeigten PSC 860, einen AT 386 als „Personal Super Computer".
Wenn es um Automatisierung, C-Komponenten und Mikroelektronik geht, spielt die weltgrößte Messe – trotz CeBIT-Konkurrenz – immer noch eine bedeutende Rolle. Die nächste Hannover Messe Industrie findet übrigens vom 10. bis 17. April 1991 statt: Allerdings finden Sie die MICROTRONIC dann nicht mehr in der Halle 13, da diese das Domizil der C-Techniken sein wird. *MP-We*

MP auf Bildungsreise

Ende Juni hatte MP gemeinsam mit etwa zehn Vertretern der DDR-Presse – Redakteuren und Mitgliedern der ehrenamtlichen Redaktionsbeiräte – Gelegenheit zu einer mehrtägigen Technologiereise. Auf dem Programm der Informationstour, zu der die Digital Equipment GmbH eingeladen hatte, standen Besuche und Besichtigungen bei verschiedenen Hard- und Softwareherstellern im südlichen Raum der BRD. Bei DEC – weltweit zweitgrößter Computerhersteller – beispielsweise im neuen Werk in Kaufbeuren, das Technologiezentrum und Fertigungsstätte für Speichertechnik ist. Hier werden gegenwärtig unter Reinraumbedingungen u. a. die Winchesterspeicher RA 92 mit 1,2 GByte Kapazität produziert. Besichtigungen des Rechenzentrums der europäischen Zentrale von EASYNET, einem der größten privaten Netze der Welt, in dem die Firma rund um den Globus etwa 50 000 Rechner verknüpft hat, zeigten die Bedeutung der Informationstechnik für moderne Unternehmen.
Bei NCR in Augsburg konnte eine nach modernsten Gesichtspunkten ausgestattete Leiterplattenfertigung und die PC-Produktion besichtigt werden. Während der PC 486/25 MC gerade in der Fertigung war, wurde die Produktion des 33-MHz-486 bereits vorbereitet. Eigenartig berührten uns in den Werkshallen sowohl bei DEC als auch bei NCR Schautafeln mit den Kennziffern der vorgegebenen und der erreichten Produktion, anspornende Losungen sowie bei NCR die Straße der „Besten des Monats" (die Fotos in Rot gerahmt). Uns kam das nur zu vertraut vor, aber es war an diesen Orten kaum zu erwarten.
Neben den Erfahrungen durch Augenschein war jedoch auch vieles zu der sogenannten Firmenphilosophie, das heißt dem Selbstverständnis der Unternehmen und der daraus abgeleiteten Marktstrategie, zu erfahren. Interessant waren hier vor allem die Schilderungen eines Mitbegründers der Firma Softlab, die 1971 trotz großer Skepsis von außen eine Softwareproduktion nach amerikanischem Vorbild begann und damals schon auf Computeranwendung zur Softwareproduktion setzte.
Heute ist die Münchener Firma unter anderem durch ihr Programmpaket Maestro und die langjährigen CASE-Erfahrungen eines der führenden Softwarehäuser der BRD.
Bei Siemens wurde ein Einblick in die Produktion von Telefonzentralen und ein Ausblick in die durch ISDN geprägte Zukunft gegeben. Schließlich gab es noch einen Besuch im Siemens-Museum in München, der auch unseren Lesern zu empfehlen ist (Eintritt frei), da hier ein Spektrum von den Anfängen der Elektrizität bis zum Megabitspeicher geboten wird.
Sowohl die insgesamt gewonnenen (auch Hintergrund-)Informationen über die besuchten Firmen als auch die Kontakte zu Kollegen der BRD-Fachpresse – freundlicherweise ebenfalls von DEC organisiert – dürften für unsere weitere Arbeit, und damit auch für Sie, liebe Leser, von nicht geringem Wert sein. *MP-We*

kennengelernt

Atari ABC 286-30



Während Atari bereits jahrelang erfolgreich bei der Entwicklung und Herstellung von Computern auf der Basis von Motorola-Prozessoren war, wurde 1986 mit dem Atari PC eine zweite Richtung eingeschlagen, die Entwicklung MS-DOS-kompatibler Technik mit Intel-Prozessoren. Der erste Atari PC verfügte über 512 KByte RAM und wurde bereits mit 8 MHz getaktet. Diese PC-Reihe wurde über den PC 2 bis hin zum PC 5 – Ataris erstem 32-Bit-PC – fortgesetzt. Etwas verwundern mag deshalb, daß mit dem ABC 286-30 die Numerierung der PC-Generationen verlassen wurde.

Der ABC – in diesem Jahr auf der Leipziger Frühjahrsmesse ausgestellt – ist wie der PC 4 ein Rechner der AT-Klasse. Seine Bestückung verrät, daß er mit einer Minimalausstattung sowohl platzsparend als auch kostengünstig konzipiert wurde (die Stellfläche seines Grundgerätes ist kaum größer als die des Monitors).

Der ABC bietet all das, was zur Zeit international zum Standard gehört: einen 80286-Prozessor, 640 KByte RAM, eine 30-MByte-Festplatte, eine batteriegepufferte Uhr sowie eine parallele und zwei serielle Schnittstellen. Der Hauptspeicher kann auf der Grundplatine (ohne daß ein Erweiterungssteckplatz benötigt wird) mit Single-inline-Memory-Modulen (SIMMs) auf 2,5 bzw. 4 MByte ausgebaut werden – abhängig vom SIMM-Typ. Von den 3 Erweiterungssteckplätzen ist einer durch die Controllerkarte für die Laufwerke belegt. Ein Sockel für den Koprozessor 80287 vervollständigt das Ganze. Als einziger Nachteil erscheint die Taktfrequenz von nur 10 MHz, die sicher den kostengünstigen Prozessor- und RAM-Schaltkreisen geschuldet ist. Eine Beschleunigung wird dagegen wiederum durch den verwendeten NEAT-Chipsatz erreicht (NEAT – New Enhanced Advanced Technology).

Standardmäßig ist der ABC mit einer 30-MByte-Festplatte und einem 3 1/2-Zoll-Floppylaufwerk mit 1,44 MByte (senkrecht in die Mitte der Frontblende eingelassen) bestückt. Darüber hinaus ist Platz für eine zweite Harddisk oder ein 5 1/4-Zoll-Floppylaufwerk mit 360 KByte. Beide können zusätzlich an die Controllerkarte angeschlossen werden. Für die Installation solcher Laufwerke wurden übrigens im mitgelieferten, deutschsprachigen Benutzerhandbuch detailliert Hinweise gegeben. Schwierigkeiten können jedoch Uneingeweihte bekommen, wenn sie ein SCSI-Laufwerk installieren wollen (übrigens nicht nur bei diesem Rechner). Sie müssen dann im Setup-Programm nicht etwa die auf dem Plattengehäuse aufgedruckte Zahl eingeben, sondern lediglich die Enter-Taste drücken. Hinter dem Punkt Fixed disk drive C type erscheint dann die Ausschrift „Not installed", was soviel heißt wie: keine Harddisk da. Nach einem Reset bootet der Rechner dann aber trotzdem anstandslos von der Platte.

Außer dem Setup-Programm bietet der Chipsatz des ABCs noch die Möglichkeit eines erweiterten Setups, mit dem die Register des NEAT-Chipsatzes beeinflußt, die Zahl der Wartezyklen voreingestellt, die DRAM-Konfiguration angegeben sowie Page- und Interleave-Modus gewählt werden. Hierfür wird das Programm NEAT aufgerufen. Nach einem Neustart greift der Prozessor mit der im erweiterten Setup eingestellten Zahl von Wartezyklen auf den Speicher zu. Der 4-Finger-Griff Control-Alt-Leftshift und + oder – auf dem Zahlenblock erlaubt aber jederzeit, von der DOS-Ebene aus zwischen 0 und 1 Wartezyklus umzuschalten. Das könnte dann sinnvoll werden, wenn der Hauptspeicher mit langsamen, (aber billigen) Speicherchips erweitert wird. Die auf der Systemplatine integrierte Grafik ist ebenfalls auf Vielfalt eingestellt. Sie unterstützt die vier Modi EGA, CGA, MDA und Hercules sowie entsprechende Farb- oder S/W-Monitore und Multisync-Monitore. Ein Umstieg auf VGA kann später durch Karten- und Monitoraustausch erfolgen.

Als Betriebssystem liefert Atari zum ABC die zur Zeit wohl am weitesten verbreitete MS-DOS-Version 3.3, und als Zusatz gibt es den GW-Basic-Interpreter (beides mit Handbuch) sowie Utilities.

Dem Anspruch, „viel Leistung für wenig Geld" zu liefern sollte Atari auch beim ABC gerecht geworden sein.

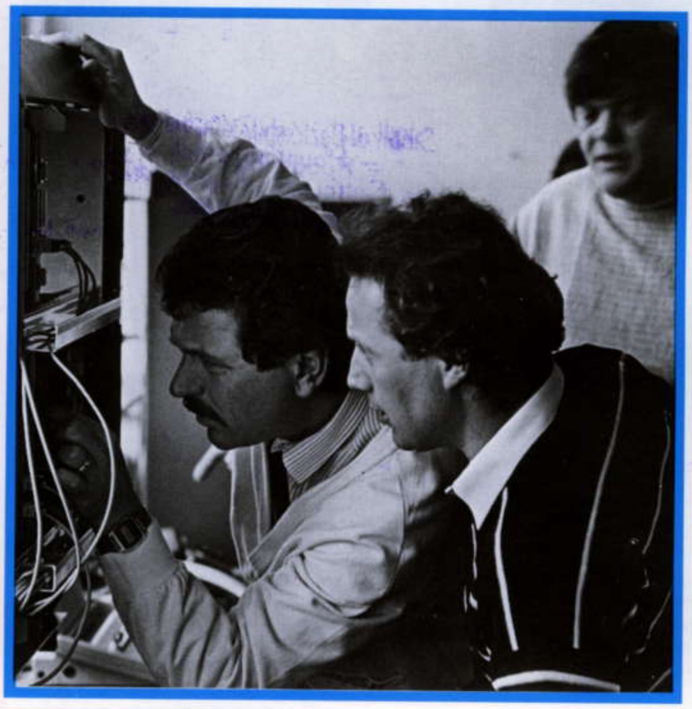
MP-Hk

Einige technische Daten

Prozessor	Intel 80286
optional	Koprozessor 80287
Taktfrequenz	8/10 MHz
RAM	640 KByte, erweiterbar auf 2,5 oder 4 MByte mit 256-KBit- bzw. 1-MBit-SIMMs
ROM	96 KByte
Schnittstellen	2 × seriell (davon 1 × für Maus), 1 × parallel
freie Steckplätze	2 × AT
Laufwerke	30-MByte-Harddisk, 5 1/4 Zoll; 1,44-MByte-Floppy, 3 1/2 Zoll; erweiterbar um eine 360-KByte-Floppy, 5 1/4 Zoll
Grafikadapter	EGA (640 × 350 Pixel, 16 aus 64 Farben) CGA (640/320 × 200 Pixel, 2/4 aus 16 Farben) MDA (720 × 350 Pixel) HGC (720 × 348 Pixel)
Monitortyp	Monochrom-, Farb- oder Multisync-Monitor
Tastatur	MF 2, 102 Tasten
Abmessungen (H × B × T)	15 × 38 × 41 cm³
Preis	2198,– Mark (8 MHz), 2298,– Mark (10 MHz)

Wir bringe
den besten
zusammen

Wir sind ein technisch-wissenschaftlicher Verband für alle Ingenieure, Naturwissenschaftler und Techniker, die in dem weiten Bereich der Elektrotechnik und Elektronik tätig sind.



Wenn Sie auch Mitglied werden wollen, senden Sie uns den ausgefüllten Aufnahmeantrag zu oder schreiben Sie einfach an die Adresse:
VDE
Verband Deutscher Elektrotechniker
Stresemannallee 15
VDE-Haus
D-6000 Frankfurt
am Main 70

Auf einen Dialog mit Ihnen würden wir uns freuen.

regional

Wir sorgen für die Weiterbildung und berufliche Profilierung. Regional ist der VDE in Bezirksvereinen organisiert. In den industriellen Zentren der DDR – beispielsweise in Dresden und Cottbus – sind bereits VDE-Bezirksvereine in Gründung. Berlin und Umgebung wird von unserem Elektrotechnischen Verein Berlin (ETV) betreut.
Informationen und Kontakte bieten wir unseren Fachkollegen in der DDR für die Zeit der Neuordnung zu Sonderkonditionen:
Beitragsbefreiung für 1990, Übergangsregelungen für 1991 und 92, verbilligter Bezug der VDE-Fachzeitschriften, vergünstigte Teilnahme an Kongressen und Seminaren.

national

Wir fördern durch unsere fünf Fachgesellschaften die Weiterentwicklung von Elektrotechnik und Elektronik auf den Gebieten Informationstechnik (ITG), Energietechnik (ETG), Mikroelektronik (GME), Meß- und Automatisierungstechnik (GMA) sowie Feinwerktechnik (FWT).
Wir erarbeiten Normen und Sicherheitsbestimmungen, prüfen elektrotechnische und elektronische Geräte und Komponenten zum Schutz und Nutzen der Verbraucher und geben technisch-wissenschaftliche Publikationen heraus.

Verband Deutscher Elektrotechniker

international

Wir haben weltweite Verbindungen. Wer heute bei wichtigen Fragen der Elektrotechnik mitreden will, braucht Weltoffenheit und internationales Format. Sie lernen auf Tagungen und Seminaren die besten Fachleute kennen.

Ich beantrage die Aufnahme als ☐ persönliches Mitglied ☐ Jungmitglied (Student)

Name | Vorname | Titel/Beruf

PLZ/Wohnort (Privatanschrift) | Straße | Telefon

Geburtsdatum | beschäftigt bei (Universität/FH-Fachrichtung – vorauss. Studienende)

PLZ/Wohnort (Geschäftsanschrift) | Straße | Telefon (bei Studenten: Heimatanschrift)

Ort | Datum | Unterschrift

Zuordnung zu einer Fachgesellschaft gewünscht? ITG – ETG – GME – GMA – FWT (max. 3)

Ich bin damit einverstanden, daß die hier gemachten Angaben für satzungsgemäße Zwecke des VDE be- und verarbeitet werden und daß dies auch mit Hilfe der elektronischen Datenverarbeitung geschieht.

Verband Deutscher
Elektrotechniker (VDE) e.V.
VDE-Haus
Stresemannallee 15

6000 Frankfurt/Main 70

ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

DM 5,–

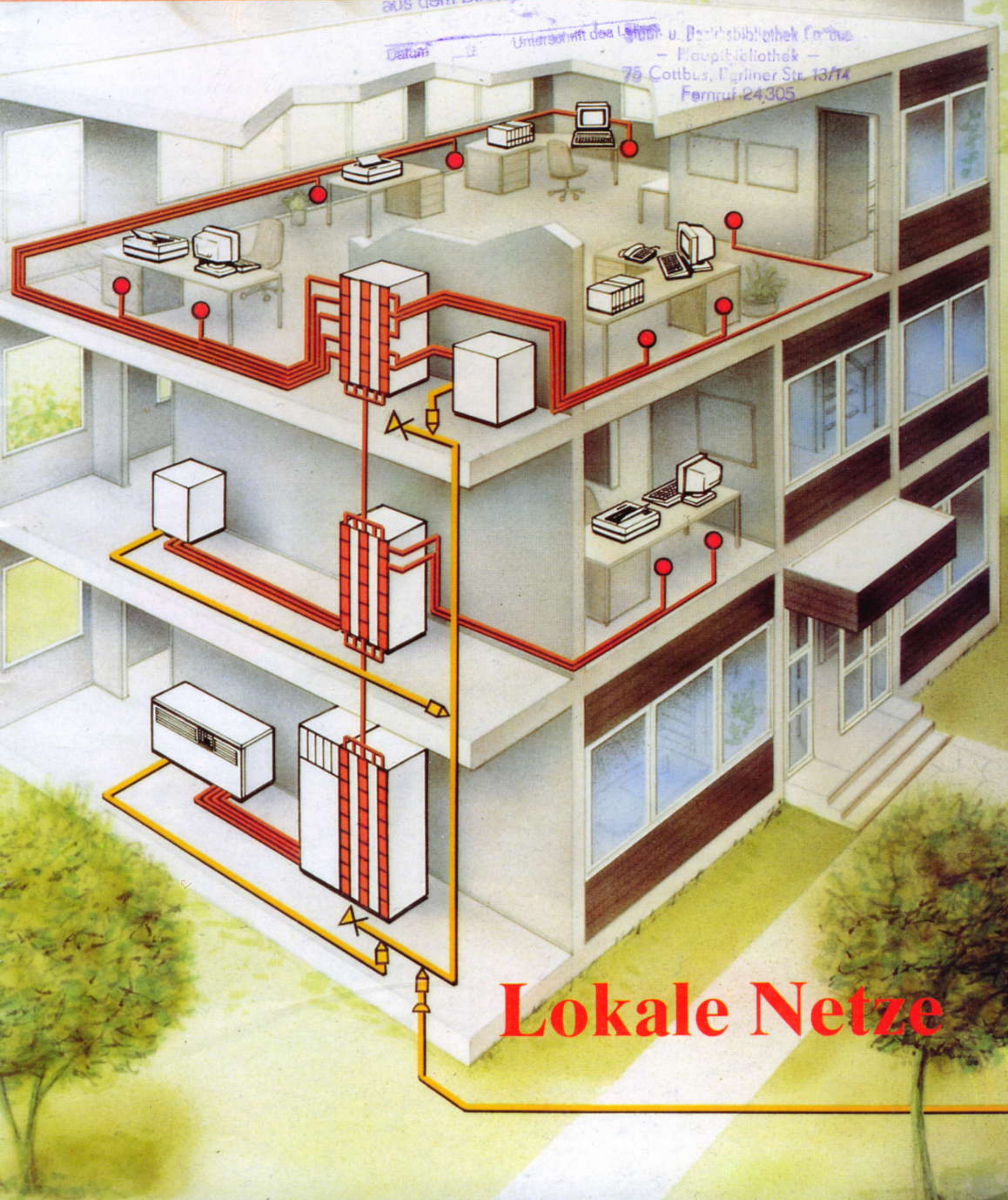
Heft 10 · 1990

Mikroprozessortechnik

Verlag Technik Berlin

ISSN 0232–2892

mit Beiträgen aus c't und Computer Persönlich



Lokale Netze

Das Beste ist gerade gut genug

Wer morgen konkurrenzfähig sein will, muß heute auf die neueste Technik setzen. Risiken kann man sich nicht leisten. Deshalb entscheiden sich große Hersteller für Produkte von **or**. Auch Sie sollten für Ihr Geld nur das Beste verlangen.

or bietet Ihnen das weltweit größte Spektrum an VMEbus Einfacheuropa-Baugruppen und das in konsequenter CMOS Technik. Das Programm umfaßt 68000 und IBM kompatible CPU-Boards, Signalprozessoren, Grafik, Massenspeicher und industrielle I/O sowie die Betriebssysteme OS-9, VRTX, VxWorks, PDOS und MS-DOS.

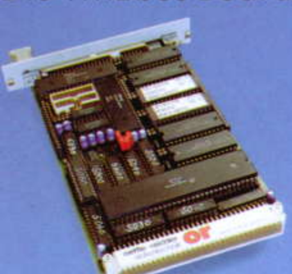
Unsere 68000 CPU Familie wurde erst jüngst auf knapp 100 Typen erweitert. Das Leistungsspektrum reicht vom 68000 bis zum 68040.

Für das Gebiet der DDR suchen wir technisch erfahrene Distributoren, Ingenieurbüros und Kooperationspartner.

or

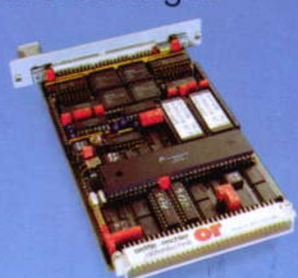
oettle+reichler GmbH
Industrial Computers
Sieglindenstr. 19½, D-8900 Augsburg 1
Tel. (0821) 5034-0, Fax (0821) 5034-119

Die VMEbus Boards für höchste Anforderungen



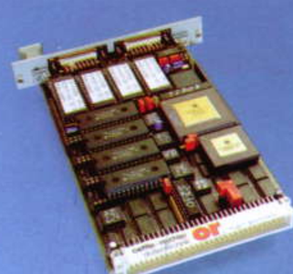
VCPU-02: Low Cost CPU Board

- 68HC000 16MHz CMOS Prozessor
- max. 2 Mbyte dual ported SRAM
- max. 4 Mbyte dual ported DRAM
- max. 1 Mbyte EPROM
- 2 async./sync. RS-232/422/485 Kanäle
- 36 TTL I/O Leitungen, interruptfähig
- Drei 16-Bit Counter/Timer
- Echtzeituhr, Watchdog
- System Controller, SGL Arbiter
- Multiprozessorfähig, Mailbox-Interrupt
- optional SCSI und Floppy Interface
- OS-9, PDOS und VRTX verfügbar
- +5V, 0,6 A, optional −40 bis +85 Grad



VCPU-00: Standard CPU Board

- 68HC000 16MHz CMOS Prozessor
- max. 512 Kbyte dual ported SRAM
- max. 1 Mbyte EPROM
- 4 async./sync. RS-232/422/485 Kanäle
- 40 TTL I/O Leitungen, interruptfähig
- Sechs 16-Bit Counter/Timer
- Echtzeituhr, Watchdog
- System Controller, 4-level Arbiter
- Multiprozessorfähig, Mailbox-Interrupt
- optional SCSI und Floppy Interface
- optional −40 bis +85 Grad
- OS-9, PDOS und VRTX verfügbar
- Low Power +5V, 0,7 A



VCPU-20: High Performance CPU Board

- MC68020 33MHz HCMOS Prozessor
- MC68881/2 Floating Point Unit
- max. 2 Mbyte dual ported SRAM
- max. 8 Mbyte dual ported DRAM
- max. 2 Mbyte EPROM, 32-Bit
- 2 async. RS-232/422 Kanäle
- Echtzeituhr, Watchdog
- System Controller, 4-level Arbiter
- Multiprozessorfähig, Mailbox-Interrupt
- optional SCSI und Floppy Interface
- optional −40 bis +85 Grad
- OS-9, PDOS und VRTX verfügbar
- Low Power +5V, 0,9 A

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
4. Jahrgang · ISSN 0232–2892

Mikroprozessortechnik, Heft 10 · 90

Verlag Verlag Technik GmbH, Oranienburger Str. 13/14, DDR – 1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd, Telefax: 2 87 02 59

Geschäftsführer Klaus Hieronimus

Redaktion Hans Weiß, Chefredakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2 87 02 03), Hans-Joachim Hill (Tel. 2870209); Sekretariat: Hannelore Kulik (Tel. 2870381)

Gestaltung Christina Bauer

Titel Werkfoto Bull AG

Beirat Prof. Dr. sc. Thomas Horn, Prof. Dr. Bernd Junghans, Dr. Dietmar Keller, Prof. Dr. sc. Bernd-Georg Münzer, Prof. Dr. sc. Peter Neubert, Prof. Dr. sc. Dr. Michael Roth (Vorsitzender), Dr. Gerhard Schulze, Prof. Dr. sc. Manfred Seifart, Dr. Dieter Simon, Prof. Dr. sc. Dr. Jürgen Zaremba

Lizenz-Nr. 1710

Gesamtherstellung Märkische Verlags- und Druck-Gesellschaft mbH Potsdam

Anzeigenannahme Verlag Technik GmbH, Anzeigenabteilung, Oranienburger Straße 13/14, Berlin, 1020; Tel. 2870254, 2870309, Telefax: 2870254

Anzeigenpreise Preisliste Nr. 1 vom 1. Juli 1990

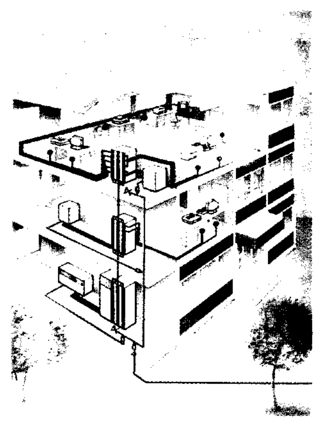
Redaktionsschluß 14. August 1990

AN (EDV) 49837

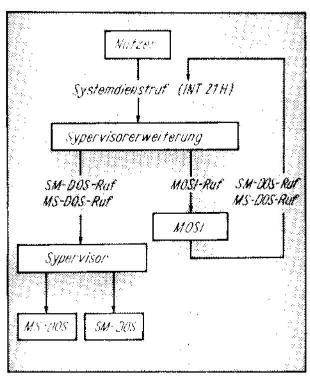
Erscheinungsweise monatlich

Heftpreis 5,– DM, Abonnementspreis vierteljährlich 15,– DM; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
Verlag Technik, Abteilung Absatz, PSF 201, Berlin, 1020 oder: *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSFR:* PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *Republik Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Republik Ungarn:* P.K.H.I., Külföldi Előfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHA-SABA, 32, Hai Ba Trung, Há Nội; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Buchhandel oder Verlag Technik GmbH



Immer mehr Firmen erkennen die Notwendigkeit, ihre Rechner zu vernetzen. Wie unser Titelbild zeigt, werden dabei verschiedenste Rechnertypen verkoppelt. Die zahlenmäßig größte Gruppe stellen dabei sicher die PCs; deshalb legt unser Titelbeitrag „Einsatz lokaler PC-Netze" auf der **Seite 4** hier den Schwerpunkt. Er gibt Ihnen einen Überblick über die Netzwerktypen und die Begriffswelt der Netze.
Ergänzend werden Ihnen mit „Starthilfe" die häufigsten Netzwerk-Fragen auf der **Seite 8** beantwortet.



Bei der Überführung von Anwenderprogrammen von einem Betriebssystem auf ein anderes ist ein Hauptproblem die unterschiedliche Parameterübergabe für die Betriebssystemrufe. Zur Überwindung dieser Schwierigkeiten standardisiert IEEE 855 – auch MOSI genannt – die Schnittstellen von Betriebssystemen. Der Beitrag „Standardgerechte Betriebssysteminterfaces" auf der **Seite 12** befaßt sich mit diesem Standardvorschlag und erläutert ein Anwendungsbeispiel für die Betriebssysteme MS-DOS und SM-DOS.

Vorschau

In MP 11/1990 müssen Sie unbedingt lesen:
- RISC
- Nantucket Tools II
- Darf's auch ein 486er sein?
- Microsoft C 6.0 PDS

Inhalt

MP-Info 2

Axel-Wüstemann, Uwe Schulze:
Einsatz lokaler PC-Netze 4

Werner Pliegl:
Starthilfe 8

Christian Hanisch:
Screen-Safer 9

Norbert Frindt, Barbara Hackler:
Standardgerechte Betriebssysteminterfaces 12

Bernd Spitzhofer:
Datenverwaltung mit dBase-kompatibler Software 14

Bernd Schewc:
Das Aids-Programm 16

Christian Hanisch:
AT-Laufwerk liest 720-KByte-Disketten 17

MP-Kurs: 19
Wolfram Schulze, Uwe Schulze:
Einführung in MS-DOS (Teil 4)

Jürgen Lang:
Zwei A 7150 an einem Drucker 23

Gert Klein:
Datenbanksysteme im Vergleich 24

Dieter Butscher:
Gründungsfieber (Teil 2) 27

Entwicklungen und Tendenzen 36

MP-Computer-Club 38
Torsten Jäkel:
Residentes Druckprogramm unter CP/M

MP-Literatur 40

vorgestellt 3. US
Neue Workstations

Technik international 4. US
Massendaten mobil

Die Elektrotechniker im vereinigten Deutschland

In Verbindung mit der bevorstehenden Vereinigung beider deutscher Staaten hat der Vorstand des Fachverbandes Elektrotechnik der Kammer der Technik (KDT) am 19. Juni 1990 beschlossen, seine Tätigkeit mit Wirkung vom 30. September 1990 einzustellen. Er geht mit seiner Entscheidung davon aus, daß in einem geeinten Deutschland die auf den Fachgebieten Elektrotechnik und Elektronik tätigen Ingenieure in einer einheitlichen Ingenieurorganisation wirken sollten und der Verband Deutscher Elektrotechniker (VDE) der Bundesrepublik Deutschland die Rolle dieser einheitlichen Ingenieurorganisation deutscher Elektrotechniker übernehmen wird.

Vorstand des Fachverbandes Elektrotechnik

Vorsitzende und Mitglieder des Vorstandes zentraler Fachgremien können laut Statut zwar als Person aus der KDT austreten, haben jedoch nicht das Recht, über die Existenz von Struktureinheiten der Kammer der Technik zu entscheiden. Dies ist ausschließlich Delegierten- bzw. Mitgliederversammlungen vorbehalten und erfordert eine Zwei-Drittel-Mehrheit.

KDT, Präsidium
Dr. Hornung, Hauptgeschäftsführer

Liebe Leserinnen, liebe Leser, viel lieber hätten wir Ihnen an dieser Stelle nur eine, dann aber für Sie eindeutige Meldung gebracht. Da dies in der gegenwärtigen „Wendezeit" nicht möglich ist, wollen wir wenigstens unserer Informationspflicht nachkommen und überlassen Ihnen die Bewertung, ohne Sie dabei zu beeinflussen. *MP*

ZVEI meint: Elektroindustrie der DDR wird wettbewerbsfähig

Die Unternehmen der westdeutschen Elektroindustrie sind seit dem Inkrafttreten der Wirtschafts- und Währungsunion verstärkt bemüht, Kontakte, Verhandlungen und Absprachen mit DDR-Elektrobetrieben in eine feste Zusammenarbeit in Produktion und Vertrieb einmünden zu lassen. In allen Sparten der Branche gibt es inzwischen Kooperationen mit DDR-Betrieben. Besonders intensiv ist die Zusammenarbeit im Infrastrukturbereich, insbesondere der Informations- und Kommunikationstechnik. Diese Meinung vertrat Prof. Dr. Rudolf Scheid, Hauptgeschäftsführer des Zentralverbandes Elektrotechnik- und Elektronikindustrie (ZVEI), anläßlich eines Pressegesprächs in Ost-Berlin.
Etwa ein Viertel der elektroindustriellen Arbeitsplätze kann nach Meinung von Scheid noch im Laufe dieses Jahres durch Kooperationsvereinbarungen in ihrer Wettbewerbsfähigkeit gestärkt werden. Diese Zahl werde sich im Verlauf des kommenden Jahres weiter erhöhen. Schließlich sei die DDR ein „naturgegeben" interessanter Standort für die westdeutsche Elektroindustrie. Zur räumlichen Nähe kämen fehlende Sprach- und Mentalitätsbarrieren sowie bestehende gute Geschäftsbeziehungen in die sich weiter öffnenden Ostmärkte und ein hohes Potential an Facharbeitern und Ingenieuren. Gerade in Anbetracht des sich verstärkenden Mangels an ausreichendem technischen Nachwuchs in der Bundesrepublik stelle das personelle Angebot der DDR ein zunehmend wichtiger werdendes Kooperationsmotiv dar. Durch entsprechende Weiterbildungsaktivitäten könnten Angebot und Nachfrage in diesem Bereich rasch zum Ausgleich gebracht werden.
Scheid rechnet damit, daß sich auf längere Frist die Strukturen der Elektroindustrie der DDR an die der Bundesrepublik annähern. Auf die heutige Ausgangssituation bezogen entspräche dies einem Umsatzvolumen von rund 50 Mrd. DM und einer Beschäftigtenzahl von knapp 300 000.

Erfinder des integrierten Schaltkreises gestorben

Am 3. Juni 1990 erlag Dr. Robert N. Noyce im Alter von 62 Jahren einem Herzschlag. Noyce war ein Pionier der Elektronikindustrie, der beispielsweise den Firmen Fairchild Semiconductor und später Intel sowie der Semiconductor Industry Association zum Leben verhalf. Bekannt wurde er jedoch vor allem dadurch, daß er gemeinsam mit Jack Kilby den ersten integrierten Schaltkreis entwickelte.
Seit 1988 wirkte Noyce in dem US-amerikanischen Sematech-Konsortium, dessen Absicht es ist, den USA wieder eine führende Position in der Halbleiterfertigung zu verschaffen.

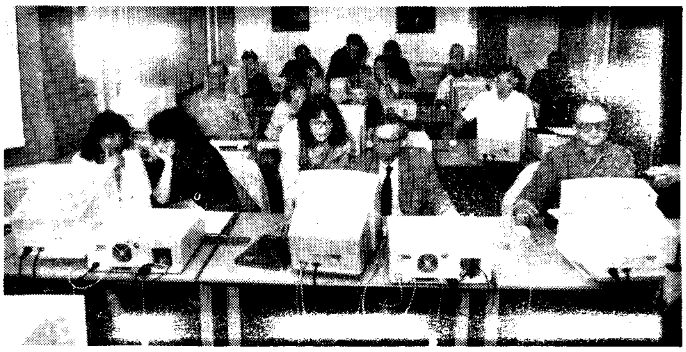
1. DDR-Unix-Netz-Arbeitstreffen

Wie beim 2. deutschen Eunet-Benutzer-Workshop am 12./13. 07 in Berlin-West im Arbeitskreis „Eunet in der DDR" angeregt, fand am 1. August 1990 das erste Arbeitstreffen zu Unix-Netzen in der DDR bei der gki* Berlin statt. Teilnehmer dieses Arbeitstreffens waren bereits aktiv am Netzgeschehen in der DDR Beteiligte. Es wurde der Stand des Aufbaus des Unix-Netzes der DDR (Xnet) besprochen, seine qualitative Verbesserung durch Nutzung des Regierungsnetzes, die angebotenen Dienste und die Bereitstellung von notwendiger Technik. Zur besseren Information aller Interessenten wird die News-Gruppe xnet.general eingerichtet. Weitere Arbeitstreffen sind vorgesehen.
Nähere Informationen erhalten Sie über: gki* Gesellschaft für offene Kommunikations- und Informationssysteme mbH, Herr Dr. Letzig, Jacques-Duclos-Straße 47–52, Berlin, 1156; Tel. 37 80 34 83

Apple gründet neue Firma

Gemeinsam mit der Apple Computer Inc. haben Bill Atkinson, Andy Hertzfeld und Marc Porat die General Magic Inc. gegründet. Das Unternehmensziel wird nach Vorstellung der Firmengründer und -leiter, die ehemalige Apple-Mitarbeiter sind, die Entwicklung einer neuen Richtung von „Personal Intelligent Communicator"-Produkten sein.
Apple wird langfristig der größte Aktionär der General Magic Inc. sein, zu der Gründung wurde jedoch lediglich ein geringer Betrag zur Verfügung gestellt. Apple besitzt weiterhin die erste, nicht-exclusiv vergebene Lizenz zur Herstellung und Vermarktung der Technologieentwicklungen. John Sculley, Chairman und CEO der Apple Computer Inc., wird Mitglied des Board of Directors der General Magic Inc. sein.
„Wir sind sehr froh, daß wir so eng mit solchen wirklichen Pionieren zusammenarbeiten können. Pioniere, deren Ziel eine revolutionär neue Sichtweise von Computing und Kommunikation ist," erläutert John Sculley das Engagement von Apple.
Dieses Projekt bot sich nach Ansicht von John Sculley deswegen an, weil das neue Unternehmen Marktsegmente ansprechen wird, die außerhalb von Apples Hauptbetätigungsfeld anzusiedeln sind.

Microsoft unterstützt das Bildungsinstitut der Presse



Um die Aus- und Weiterbildung von Journalisten, die verstärkt die Schreibmaschine gegen einen PC eingetauscht haben, auf der Basis moderner Standardsoftware sicherzustellen, rüstet Microsoft das im April gegründete Bildungsinstitut der Presse (München/Ostberlin) mit seinen aktuellsten Applikationen aus. Über das vom ehemaligen Verband der Journalisten der DDR (VDJ) und dem Verband deutscher Nachwuchsjournalisten (VdN) initiierte Institut soll Medienvertretern der praxisorientierte Umgang und die Arbeit mit dem integrierten Softwarepaket Microsoft Works, dem Tabellenkalkulationsprogramm Microsoft Excel, der Textverarbeitungssoftware Microsoft Word und Microsoft Word für Windows sowie der grafischen Bedieneroberfläche und Betriebssystemerweiterung Microsoft Windows nahegebracht werden.
Mit der Schulung von 15 Journalisten und Pressegestaltern aus der DDR, die am 15. Juni 1990 eine erste Einführung in die wichtigsten Software-Kategorien vorsah und den Teilnehmern einen grundlegenden Überblick im Umgang mit dem PC im „journalistischen Alltag" gab (siehe Bild), hat die Unterstützung durch das Softwarehaus bereits konkrete Formen angenommen.

Robotron-Softwaremesse 1990

Wir waren zwar nicht eingeladen, aber wir sind trotzdem hingegangen. Und weil keine Feen, sind wir auch nicht nachtragend ...
Robotron-Projekt Dresden, ein Softwarebetrieb des ehemaligen DDR-Computermonopolisten, zeigte zum wahrscheinlich letzten Mal auf einer eigenen Softwaremesse eigene Softwareprodukte. Austragungsort der 3 1/2-Tage-Veranstaltung war Dresden, und der Termin (17. bis 19. April 1990) lag günstig – zwischen zwei Hannover-Messen. Neu gegenüber den vorangegangenen Dresdner Messen war die hohe Beteiligung bundesdeutscher Softwarefirmen, die viele Produkte anboten, die, nicht zuletzt dank Robotron, vielen der anwesenden DDR-Nutzer, meist zwar in etwas älteren Versionen, so doch aber bekannt waren.
Und so kamen Microsoft, Ashton-Tate, Wordstar, Oracle, aber auch der Verein zum Schutz der Softwareindustrie. Siemens, Commodore, Lotus Development und Word Perfect waren da und auch andere, aus den bisher so verbündeten RGW-Staaten (Videoton/Ungarn, Algoritm und INEUM/SU, INSAG/Kuba, Interprogramm/Bulgarien).
Von besonderem Interesse dürfte für viele Textverarbeiter die erste öffentliche DDR-Aufführung von Wordstar in original Bild und Ton gewesen sein, dazu noch in der neuesten Version 5.5. Und nebenan, auch mit neuesten Version 5.1: Word Perfect. In der MP wollen wir beide Produkte in der nächsten Zeit etwas ausführlicher vorstellen. *MP*

Erste Mitgliederversammlung des UVI

Am 27. Juni 1990 fand im Zentralinstitut für Kybernetik und Informationsprozesse der Akademie der Wissenschaften der DDR die erste Mitgliederversammlung des Unternehmensverbandes Informationssysteme e.V. (UVI) statt.
Viele Informatiker und Teams mit jahrelanger Erfahrung und hoher Kreativität, die sich unter den unzureichenden technischen Voraussetzungen bewähren mußten, sehen jetzt eine Chance für die Gründung eigener Unternehmen. Sie leisten damit unter persönlichem Risiko einen Beitrag zur Schaffung einer mittelständischen Industrie auf dem Territorium der DDR.

Ein Defizit der technisch sehr versierten und hoch motivierten Jungunternehmer liegt jedoch nach ihrer eigenen Einschätzung in der Betriebswirtschaft, im Management und im Marketing. Darüber hinaus erlauben es die finanzielle Situation und die geringe Mitarbeiterzahl vieler neuer Unternehmen noch nicht, eigene Entwicklungen anzugehen, was langfristig gesehen die Innovativität als dauerhafte Existenzgrundlage beeinträchtigt.
Zur Lösung vieler sich daraus ergebender Aufgaben eignet sich die Organisationsform eines Unternehmensverbandes, die sich auch unter den stabilen marktwirtschaftlichen Verhältnissen in der BRD bewährt hat.
Die Initiative zur Gründung eines solchen Verbandes in der DDR entstand in der Seminarreihe „Softwareunternehmen – Strategien für den Markt", die seit Anfang des Jahres 1990 von einer Projektgruppe am Institut für Theorie, Geschichte und Organisation der Wissenschaft der AdW durchgeführt wird. In diesen Seminaren beraten Teilnehmer aus Industrie und Forschung über die Chancen und Risiken der DDR-Anbieter von Informationssystemen auf dem deutschen und internationalen Markt.

Ziele und Aufgaben des UVI e.V.
Auf der ersten Vollversammlung des am 6. Juni gegründeten Unternehmensverbandes Informationssysteme stellte der Vorsitzende des Vorstandes, Herr Dipl.-Ing. J. Pialek, die Ziele, Aufgaben sowie die Arbeitsweise des Verbandes dar, der am 26. Juni in das Vereinsregister eingetragen wurde. Er ließ keine Zweifel daran, daß es sowohl eines außergewöhnlichen Engagements in den neu gegründeten und umstrukturierten Unternehmen als auch der gegenseitigen Unterstützung bedarf, um unter den internationalen Anbietern auf dem Markt für Informationssysteme zu bestehen.
Der UVI e.V. will die Interessen der auf dem Territorium der DDR tätigen Unternehmen und Institutionen vertreten, die Produkte und Dienstleistungen für Informationssysteme anbieten. Er soll sich zu einem Kommunikationszentrum der Branche entwickeln und Ansprechpartner für die Landesregierungen, internationale Institutionen und Kooperationspartner aus dem Ausland werden. Besonderes Anliegen des UVI e.V. ist die umfassende Erstberatung und Unterstützung von Firmengründungen.

BRD-Verbände als Partner
In der BRD gibt es mehrere Verbände im Bereich der Datenverarbeitungsindustrie. Einige von deren führenden Vertretern nahmen an der Vollversammlung des Unternehmensverbandes Informationssysteme e.V. teil und bestärkten die Initiatoren in der Weiterführung der Arbeit, vermittelten Ideen zur Verbandsarbeit und unterbreiteten Vorschläge zur „Kooperation statt Konfrontation".

Seminare im Herbst
Zu den nächsten Aktivitäten des Verbandes gehören drei ganztägige Weiterbildungsseminare im Herbst, die gemeinsam mit bundesdeutschen Partnern vorbereitet wurden bzw. werden. Am 24. Oktober geht es um Versicherungsschutz für Unternehmen der Informationsverarbeitung, am 28. November stehen der Software-Rechtsschutz und am 7. Dezember das Controlling in Softwareprojekten auf dem Programm. Weitere Veranstaltungen sind am 16. Januar 1991 zur staatlichen Förderung innovativer Unternehmen der Informationsverarbeitung und am 6. Februar 1991 zum Marketing in diesen Unternehmen.

Interessenten wenden sich bitte an die Geschäftsstelle des UVI e.V., Dr. sc. Mathias Weber, Institut für Theorie, Geschichte und Organisation der Wissenschaft, Akademie der Wissenschaften der DDR, Prenzlauer Promenade 149–152, Berlin, 1100; Tel.: 4 79 72 75/2 78 *H. Belitz*

Amiga-'90-Messe

Nachdem die erste europäische Amiga-Messe 1989 mehr als dreimal soviel Besucher wie erwartet verzeichnen konnte, wird für die dritte Messe diesmal in zwei Hallen eine etwa fünfmal so große Ausstellungsfläche zur Verfügung stehen. Austragungsort ist wiederum die Kölner Messe, nachdem die zweite Amiga-Messe in der Schweiz stattfand. Bis Anfang August hatten sich bereits 116 Aussteller angemeldet; unter ihnen natürlich als Schirmherr der Hersteller des Amiga, die Firma Commodore, sowie das Amiga-Magazin. Während sich in der Halle 10.2 die Profis über Hardware und professionelle Anwendersoftware informieren können, ist die Halle 12.2 den Unterhaltungsmöglichkeiten des Amiga gewidmet. Das heißt, Spiele, Musiksoftware und Hardware können hier erworben, Versandhändler und Verlage aufgesucht werden. Ergänzt wird die Messe wieder durch Seminare, für die internationale Dozenten zur Verfügung stehen.
Die Amiga '90 findet vom 9. bis 11. November 1990 statt und ist täglich von 9 bis 18 Uhr geöffnet; vorgeschaltet ist der Messe ein Fachbesuchertag.
Der Tages-Eintrittspreis beträgt für Schüler und Studenten 10,– DM, für Erwachsene 15,– DM.

Fachmesse Computergrafik

Das Zentrum für graphische Datenverarbeitung Darmstadt (ZGDV) und die Universität Rostock sind seit vielen Jahren auf dem Gebiet der Computergrafik als angewandte Wissenschaft tätig. Als Auftakt zur Gründung eines Demonstrations- und Qualifizierungszentrums Rostock veranstalten die beiden Institutionen am 29. und 30. November 1990 in Rostock eine Fachmesse Computergrafik. Zur Förderung fachlicher Beziehungen zwischen Hochschulen und Industrie werden moderne Hard- und Software, integrierte Systemlösungen und Technologien sowie aktuelle Fachliteratur vorgestellt. Daneben gibt es Fachvorträge und eine Podiumsdiskussion zum Thema Computergrafikausbildung. Angesprochen werden als Zielgruppe vor allem DDR-Betriebe, Jungunternehmer und Ingenieure, Wissenschaftler, Techniker sowie private Computeranwender.
Kontakt: Universität Rostock, Sektion Informatik, Rostock, 2500.

PCs im Versandhandel

In diesen Tagen erreichte der aktuelle Herbst-/Winterkatalog 1990/1991 von Neckermann fast eine Million Haushalte in der DDR. Mit der Möglichkeit, über den bequemen Bestellweg die gewünschte Ware auswählen, zu Hause prüfen und bei Nichtgefallen innerhalb von 14 Tagen an die Neckermann-Zentrale zurücksenden zu können, ist dieser Weg vor allem für private Computerkäufer sicher eine interessante Variante. Bekanntlich umfaßt das Katalogangebot bewährte Markenartikel des PC-Sektors unter anderem von Atari, Commodore und Schneider. Daneben eröffnete der Neckermann Versand im Juli in Cottbus, Dresden und Gera in den Räumen des ehemaligen Intershop Verkaufshäuser für das Sortiment TechnikWelt, die eine fachgerechte und kundenfreundliche Präsentation erlauben. Sie ergänzen den technischen Kundendienst „Rat + Tat", der bereits in den Centrum-Warenhäusern in Dresden, Erfurt, Görlitz, Halle, Leipzig, Magdeburg und Suhl zur Verfügung steht. Bestellungen und Anfragen kann man jedoch auch an die Neckermann Versandhandelsgesellschaft mbH, Paradiesstraße 210, Berlin, 1183, richten.

Produktion von Siemens-Rechnern in der DDR

Der Bereich Daten- und Informationstechnik der Siemens AG, Berlin und München, und die neugegründete Computer Elektronik Dresden GmbH (CED) vereinbarten in Rahmenverträgen eine Zusammenarbeit in Hardware-Entwicklung und Fertigung. Ab 1992 soll ein jährliches Fertigungsvolumen von rund 400 Mio DM erreicht werden.
Die CED GmbH, ehemals Robotron Elektronik Dresden (RED) wird dabei Aufträge zur Hardware-Entwicklung ausführen. Die Produktion von Siemens-Rechnern der Linien MS-DOS, Sinix und BS2000 läuft unter Sicherstellung der hohen Qualitätsmaßstäbe im Oktober in Dresden an. Es werden Personalcomputer PCD-2, Sinix-Mehrplatzcomputer MX 300 und die Zentraleinheit 7.500-H60 in Lizenz gefertigt. Zunächst wird mit der Hochrüstung von Rückläufern der Zentraleinheiten 7.560/BS2000 begonnen.
Erste Auslieferungen sind für Ende 1990 geplant. Der Verkauf der Produkte wird über die bestehende Siemens-Vertriebsorganisation bzw. neue Vertriebsgesellschaften für die DDR, die Sowjetunion und andere Länder Osteuropas erfolgen.

PC-Markt im Abwärtstrend?

Nach jetzt vorliegenden Analysen des Marktforschungsunternehmens IDC wurden in der BRD 1989 gegenüber 1988 fast 19 000 Desktop-Personalcomputer weniger verkauft. Insgesamt waren es 1,44 Millionen PCs. (Bei dieser Gelegenheit revidierte IDC die 1989 getroffene Einschätzung des Jahrganges 1988 übrigens dahingehend, daß nicht, wie damals angenommen, die 1,5-Millionen-Grenze überschritten worden war, sondern knapp 1,45 Millionen PCs verkauft wurden.) Bei einer Differenzierung läßt sich erkennen, daß der Rückgang eindeutig zu Lasten des Heimcomputersektors geht. Hier hatte vor allem die Firma Commodore durch jetzt nachlassendes Interesse am C 64 und C 128 Rückgänge zu verzeichnen. Allerdings liegt sie mit 29,6 Prozent der verkauften Geräte noch in Spitzenposition (Atari 13,9 %, IBM 9,2 %, Schneider 6,0 %). Der Markt für die billigen Heimcomputer ist nach Meinung von IDC offensichtlich gesättigt; dafür spricht auch der drastische Rückgang der 8-Bit-Rechner von 32,4 % auf 19,2 %. Nach den wichtigsten Prozessoren der PCs aufgeschlüsselt ergeben sich folgende Werte: Motorola 680x0 von 22,6 auf 20,4 % gefallen, 8086/8088 von 18,9 auf 17,9 % gefallen, 80286 von 21,9 auf 33,7 % gestiegen, 80386 von 4,2 auf 8,8 % gestiegen. Hier zeigt sich, daß der Trend zu den leistungsstärkeren PCs geht, wobei die Nachfrage für den Business-Bereich außerordentlich steigt. So wurden in diesem Segment 1989 nicht nur die vorausgesagten 560 000 Stück erreicht, sondern 720 000. Die Frage in der Überschrift darf man demnach wohl mit einem klaren Nein beantworten.
Ganz nebenbei: Trotz des geringeren Absatzes 1989 stieg der Umsatz um 20 Prozent, was auf den überdurchschnittlichen Verkauf der leistungsstärkeren und teureren PCs zurückzuführen ist. So gesehen dürfte es für die Hersteller also kaum Grund zur Klage geben.

Kostenlose Gast-Mitgliedschaft im NCG-Verein

Um einen raschen und intensiven Informationsfluß über die Anwendung neuer Technologien in der industriellen Fertigung zu ermöglichen, können vor allem kleine und mittlere Unternehmen aus der DDR eine kostenlose Gast-Mitgliedschaft für 1990 und 1991 gewährt bekommen. Dieses Angebot gilt insbesondere für Unternehmen, die sich mit der Anwendung, das heißt mit dem Einsatz von CNC-Technologien und CAD/CAM-Technik, befassen oder damit beginnen wollen. Die Gast-Mitgliedschaft berechtigt zur Teilnahme an allen NCG-Veranstaltungen zu Mitgliederkonditionen, Mitarbeit in NCG-Arbeitskreisen, Bildung neuer Arbeitskreise usw. Die Gast-Mitgliedschaft beinhaltet jedoch kein satzungsmäßiges Stimmrecht im NCG-Verein.
Interessierte Unternehmen können Anmeldeunterlagen bei der Geschäftsstelle des NCG-Vereins, Postfach 31 68, 7900 Ulm, anfordern.
Am 13. und 14. Juli veranstaltete der NCG-Verein übrigens vor 190 Fachleuten in Chemnitz sein 1. Mitgliederforum in der DDR; das 2. Forum wird am 2. und 3. November 1990 in Dresden stattfinden. Interessenten wenden sich bitte an die vorgenannte Anschrift.

Drei Jahre Garantie

Als erster PC-Hersteller stellte die Firma a & m Computertechnik aus München für ihre Produkte zur CeBIT 90 ihr Servicekonzept vor, das eine 3jährige Garantie mit kostenlosem Vor-Ort-Service beinhaltet. Diese ursprünglich für das Gebiet der BRD gebotene Dienstleistung wird jetzt, nachdem geeignete Kooperationspartner gefunden wurden, auch den Käufern in der DDR geboten.

Einsatz lokaler PC-Netze

Axel Wüstemann, Uwe Schulze, Rostock

Wer ein lokales Netz aufbauen oder auch nur (unter Ausnutzung aller Möglichkeiten) betreiben will, muß schon einiges an Sachkenntnis mitbringen: Zu viele Standards und Begriffe gilt es zu verstehen, zu verschieden sind die angebotenen Produkte. Denn schließlich soll zum Schluß alles zusammenpassen. In diesem Beitrag werden einige Grundlagen und Entscheidungshilfen vermittelt.

In puncto Leistungsfähigkeit stehen PCs der oberen Leistungsklasse vielen ihrer – nicht nur vom Namen her – größeren Kollegen nur wenig nach. Was noch fehlte, waren die Möglichkeiten der gemeinsamen Nutzung von Daten und Ressourcen. Die externen Speicherkapazitäten der Großrechner treten eher in den Hintergrund, denn auch PCs lassen sich inzwischen mit Festplatten und Streamern von mehreren 100 MByte ausrüsten. An dieser Stelle soll nicht die allenthalben in der Fachpresse geführte Diskussion PC-Netz versus Mehrplatzsystem fortgesetzt werden. Beide Formen gemeinsamer Rechnernutzung werden auch in Zukunft nebeneinander bestehen und besitzen ihre Berechtigung: Zum einen steht die Aufgabe, Verbindungen zwischen bestehender Technik zu schaffen, zum anderen wird auch in Zukunft verschiedenartige Technik eingesetzt werden – abhängig vom Preis-Leistungs-Verhältnis und der Eignung für den Einsatzzweck (Büroautomatisierung, Prozeßsteuerung etc.). Welcher Stellenwert der Verbindung von Produkten verschiedener Anbieter zukommt, ließ sich auch an der CeBIT '90 ablesen, wo alle namhaften Hersteller (darunter IBM, Hewlett Packard, Schneider & Koch) unter dem Motto Connectivity 90 ihre Produkte in einem großen Verbund zeigten. Der immer größer werdenden Produktvielfalt wirken also auch Bestrebungen nach Vereinheitlichung – zumindest aber nach Verbindung – entgegen. Besondere Bedeutung erhalten Übergänge zwischen verschiedenartiger Technik durch das große Angebot an Endgeräten (Telex, Telefax, hochwertige Drucker, Fotobelichter, Modems, paketvermittelte Datennetze), die ortsunabhängig von allen Arbeitsstationen aus genutzt werden können. So haben bei uns im Hause DOS- und Unix-Nutzer gleichermaßen Zugriff auf den Laserdrucker.

Die ersten PC-Netze dienten zunächst vor allem dazu, durch eine gemeinsame Nutzung von teuren Peripheriegeräten Kosten zu sparen. Mit der 1983 erfolgten Markteinführung von Novells Netzwerkbetriebssystem NetWare und der Standardisierung von Netzwerkfunktionen im MS-DOS ab der Version 3.0 rückte die gemeinsame Nutzung von Daten in den Vordergrund. Mit einer breiten Palette von netzwerkfähigen Anwendungen, wie etwa dBase ab Version III Plus, ist es möglich, auf dem Datei-Server verwaltete Daten arbeitsteilig durch mehrere Benutzer im Netzwerk gleichzeitig zu nutzen. Eine echte Integration verschiedener DV-Welten, wie sie gerade in größeren Unternehmen historisch gewachsen sind, ist mit dieser Technologiestufe jedoch nur bedingt möglich. Mit der Einführung von NetWare 386 oder dem OS/2-LAN-Manager 2.0 wird auch diese Aufgabe in Angriff genommen. Durch die Möglichkeit, verschiedene Übertragungsprotokolle zu nutzen, sowie durch den Übergang zu verteilten Client-Server-Anwendungen kann dieser Schritt vollzogen werden.

Grundlage für die Architektur von Netzwerken ist das *Open-System-Interconnection*-(OSI-)Referenzmodell der Internationalen Standardisierungsorganisation (ISO). In Bild 1 ist es übersichtsartig dargestellt, nebst einer Einordnung der verschiedenen in diesem Artikel erwähnten Protokolle.

Für den Praktiker ist das OSI-Referenzmodell ein gutes Hilfsmittel, die verschiedenen Kommunikationsprodukte, Protokolle usw. einzuordnen und zu bewerten. Insbesondere bei der Integration verschiedener Systemarchitekturen hilft das Referenzmodell, Klarheit zu schaffen und Entscheidungen zu treffen.

Die durch die ISO für die einzelnen Schichten definierten Protokolle und Dienste halten in kommerziellen Produkten nur langsam Einzug. Das trifft insbesondere auf PC-Netze zu. Lediglich die Schichten 1 und 2 werden durch die meisten Hersteller entsprechend den Standards realisiert. Oberhalb der Schicht 2 existiert eine Vielzahl firmenspezifischer Protokolle, die sich zum Teil als Industriestandards etablieren konnten.

Industriestandards

In der Praxis haben sich drei Netze etabliert: Ethernet (mit der Variante Cheapernet), IBM Token Ring und Arcnet. Die weiteste Verbreitung im PC-Bereich weist das **Ethernet** auf, das im Xerox Palo Alto Research Center entwickelt und frühzeitig von Intel und DEC unterstützt wurde. Es benutzt das Zugriffsverfahren CSMA/CD (Carrier Sense Multiple Access with Collision Detection). Der komplizierte Name steht für ein einfaches Verfahren, bei dem der Sender durch Abhören des Netzes auf einen Zeitpunkt wartet, zu dem keine Daten über die Leitung gehen. Um einem gleichzeitigen Sendebeginn zu begegenen, wird anschließend auf Überlagerungen des eigenen Signals mit dem anderer Stationen geprüft und die Übertragung gegebenenfalls später neu gestartet. Bei diesem Verfahren ist mit CD die Kollisionserkennung gemeint. CSMA/CD ist als IEEE 802.3 und ISO 8802.3 genormt, Ethernet unterscheidet sich aber im Paketaufbau geringfügig davon. Das verwendete Koaxialkabel ist eine Spezialentwicklung und wird auch als **Yellow Cable** (aufgrund seiner gelben Farbe) oder **Thick Wire** bezeichnet, was auf den großen Durchmesser hindeutet, weshalb die Verkabelung auch nicht ganz einfach ist. Der Vorteil liegt aber in den großen Segmentlängen (bis zu 500 m), weshalb es oft als sogenanntes Backbone-Netz, also zur Zentralverkabelung von Gebäuden, verwendet wird. Der Anschluß des Rechners an das Ethernet-Kabel erfolgt über Transceiver (Kunstwort aus Transmitter und Receiver, also Sender und Empfänger), von wo aus ein spezielles 15poliges Transceiverkabel (Drop Cable) zum Rechner führt. Der Transceiver erfüllt die Funktionen der Zugriffssteuerung auf physischer Ebene. Die eigentliche Ethernet-Controller-Karte kann recht verschieden ausgeführt sein. Ihre Leistungsfähigkeit wird von

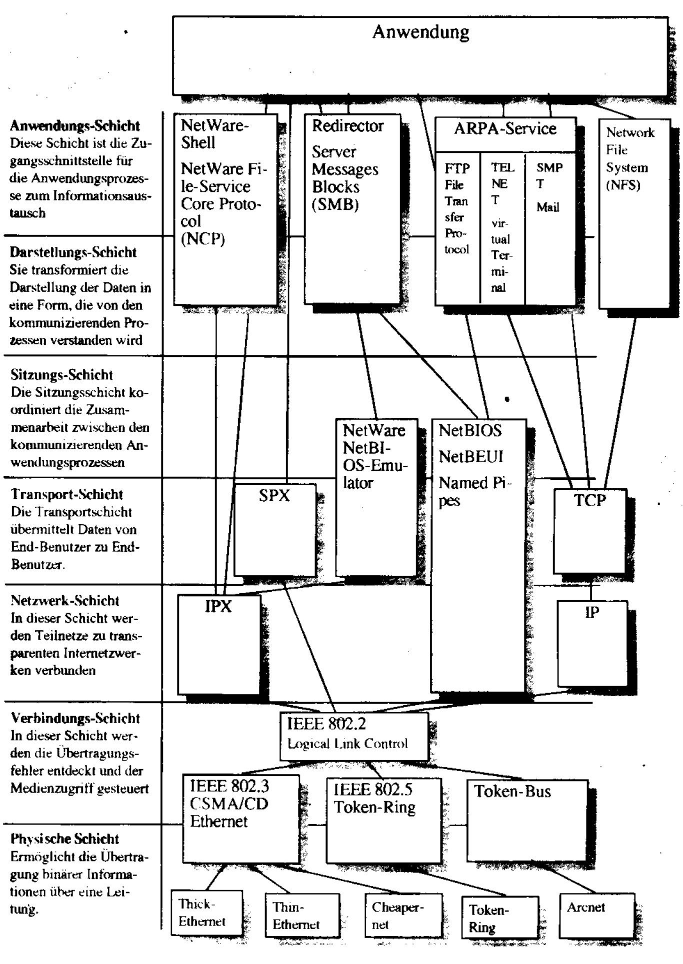


***Bild 1** Das OSI-Referenzmodell der ISO in der Praxis*

der Busbreite (8, 16 oder 32 Bit), von der RAM-Größe (für die Datenpufferung) und von der Intelligenz (Spezialprozessoren für die Codierung, CRC-Prüfung etc.) bestimmt.
Die relativ hohen Kosten für ein Ethernet und die damit verbundene komplizierte Verkabelung waren Anlaß für Überlegungen, statt des teuren Ethernet-Equipments Standard-Koaxial-Kabel zu verwenden, was ohne weiteres möglich ist, da die Übertragungsrate nicht durch das Kabel vorgegeben ist, sondern durch die definierte Bit-Zeit der Manchester-Codierung (100 ns). Die etwas schlechteren Eigenschaften bezüglich des Dämpfungsverhaltens bedingen allerdings kürzere Segmentlängen. Das dünnere Kabel wird (den günstigeren Preis kennzeichnend) **Cheapernet** genannt. Tafel 1 vergleicht beide Netze. Die wesentlich kleineren Transceiver befinden sich direkt auf den Controller-Karten, so daß weitere kostenintensive Komponenten entfallen.
Häufig werden in einem großen Netz Ethernet und Cheapernet durch Kopplung über Repeater kombiniert. Nach einer Marktstudie sind Netzwerke auf Ethernet-Basis (einschließlich Cheapernet) mit 72 % am Umsatz in diesem Marktsegment beteiligt.
Der **IMB Token Ring** ist ein 1985 entwickeltes und von IBM maßgeblich gefördertes Netzwerk, das deren gesamte Produktpalette einschließt (Mainframes, Superminis, PCs). Beim (schon länger bekannten) Token-Passing-Verfahren wird ein Steuerrahmen (eben das Token, was sich mit Staffelstab übersetzen läßt) permanent durch das Netz geschickt. Solange es leer ist, kann der Sender seine Daten einspeichern. Der Empfänger liest sie aus und kennzeichnet das im Token. Kommt das Token wieder beim Absender an, so liest dieser die Quittung und leert anschließend den Rahmen. Durch dieses deterministische Zugriffsverfahren werden Kollisionen und deren Behandlung vermieden. Allerdings ist der Aufwand für eine Ringverkabelung um einiges größer als das beim Bus der Fall ist, zumal der IBM Token Ring genaugenommen als Stern verkabelt ist, da das Kabel von jeder Station zu einem sogenannten Ringverteiler zurückgeschleift wird. Dieses scheinbar umständliche Verfahren ist notwendig, um das Netz auch bei Ausfall einer Station lauffähig zu halten, da das Token von jeder Station neu generiert werden muß. Die weitere Entwicklung des Token Rings führt zu einer Version mit 16-MBit-Übertragungsrate, wobei zu sagen ist, daß die Bruttoraten verschiedener Netze nur bedingt vergleichbar sind, da sich aufgrund des unterschiedlichen Protokolloverheads auch verschiedene effektive Übertragungsraten ergeben.
Eine alternative Entwicklung zu den beiden beschriebenen Netzen ist **Arcnet** (Atteched Resource Computer Network) der Firma Data Point Corporation, das 1977 auf den Markt kam. Während Ethernet und Token Ring anfangs recht teuer und aufwendig zu verkabeln waren, eroberte sich das billige Arcnet schnell den PC-Markt. Wenngleich nicht in die Normungen der internationalen Institutionen eingegangen, handelt es sich doch um einen am Markt etablierten Industriestandard. Man kann in ihm den Versuch der Verbindung der Vorteile von Ethernet und Token Ring sehen: Auf einer kostengünstigen Bus-Struktur wird ein deterministisches Token-Verfahren eingesetzt. Da Arcnet nur relativ kurze Entfernungen überbrücken kann, ist der Einsatz von Verteilern – Hubs genannt – nötig, die wiederum kaskadiert werden können, so daß sich eine Baumstruktur ergibt. Arcnet-Karten werden auch für einfache, verdrillte Leitung und für Lichtleiterkabel angeboten. Das neue Arcnet PLUS bietet eine Übertragungsgeschwindigkeit von 20 MBit/s.
Das Hochleistungsnetz der Zukunft wird **FDDI** (Fibre Distributet Data Interface) darstellen. Es benutzt als Verkabelung zwei gegenläufige Ringe aus Glasfaserkabel und als Zugriff ein optimiertes Token Passing Verfahren. Damit wird eine ausgesprochen hohe Geschwindigkeit erreicht (100 MBit/s). Die mögliche große räumliche Ausdehnung (bis 100 km) und Stationszahl (bis 500 Stationen), aber auch die hohen Kosten lassen FDDI besonders für Backbone-Netze geeignet erscheinen. Tafel 2 stellt die verschiedenen Netze gegenüber.

Tafel 1 Vergleich Ethernet und Cheapernet

	Ethernet	Cheapernet
maximale Segmentlänge	500 m	185 m
maximale Segmentzahl	10	10
maximale Ausdehnung	5 km	1,85 km
Transceiver-Abstand, mindestens	2,5 m	1 m
Länge Transceiver-Kabel, maximal	50 m	–
Kabel	Ethernet Koaxial	RG-58
Transceiver	extern	on board

Tafel 2 Die wichtigsten Merkmale verschiedener Netze

	Ethernet	Arcnet	Token Ring	FDDI
Übertragungsrate	10 MBit/s	2,5 MBit/s	4 MBit/s	100 MBit/s
Bereich	bis 2,5 km	bis 6,5 km	einige km	bis 20 km
Medium	Ethernetkoaxkabel, RG-58 (für Cheapernet)	Koaxkabel RG-62	Zweidraht	Lichtleiter
Protokoll	CSMA/CD	Token Bus	Token Ring	Token Ring
Topologie	Bus	Baum	Ring	Doppelring

Protokollschnittstellen

Im Gegensatz zu den Zugriffsverfahren, die vorwiegend technisch bedingt sind, gehen die unterschiedlichen Protokolle auf die Entwicklungen verschiedener Hersteller zurück. Ihre Stärken zeigen zumeist auch die Ziele der Entwicklung: größtmögliche Geschwindigkeit, universeller Einsatz oder Internetzwerkfähigkeit. Zumeist sind sie an die Netzbetriebssysteme oder andere Software des jeweiligen Entwicklers gebunden. Einige Protokolle haben sich aber so weit als Standard etabliert, daß ihnen auch Fremdanbieter folgen.
Ein Übertragungsprotokoll regelt den Dialog zwischen zwei Teilnehmern. Daten werden nicht Byte für Byte verschickt, sondern in **Paketen** (z. B. von 256 Byte). Diesen müssen Informationen zum Absender und Empfänger sowie eine Prüfsumme beigefügt werden. Außerdem sind Regelungen zu treffen, wie im Fehlerfalle zu verfahren ist. Beispielsweise kann der Absender ein Paket wiederholt abschicken, wenn in einem bestimmten Zeitraum keine Quittung eintrifft. So brauchen fehlerhafte Pakete einfach nicht quittiert zu werden. Eine andere Möglichkeit ist, daß die wiederholte Absendung von Paketen explizit angefordert wird; allerdings ist es so schwieriger, verlorengegangene Pakete zu ermitteln. Zusätzlicher Aufwand entsteht, wenn in einem großen Netz mehrere Wege zwischen Absender und Empfänger existieren. Hier muß zusätzlich auf die Paketreihenfolge geachtet werden.
Die wichtigsten Protokolle im PC-Bereich sind NetBIOS (IBM), IPX (Novell) und in zunehmenden Maße auch TCP/IP, das aus der Unix-Welt stammt. Alle drei unterstützen Internetworking, können also Adressen verschiedener Netze bearbeiten und damit netzübergreifend eingesetzt werden.
Mit der Verbreitung lokaler Netze wuchs der Wunsch, einen Netzzugang über die bewährten Betriebssystemschnittstellen zu schaffen. Der Gedanke lag nahe, gerade angesichts der vielfältigen Netzhardware einen ähnlichen hardwareunabhängigen Aufruf für ein virtuelles Netzwerk-Gerät zu gestalten, wie für die traditionelle DOS-Peripherie. Mit Einführung ihres ersten Netz-Adapters (des IBM PC-Network-Adapters) erweiterte IBM deshalb sein BIOS um eine Netzwerkkomponente, das NetBIOS.
Meist liefert der Hardware-Hersteller ein **NetBIOS** als COM-Datei zur Netzwerkkarte mit. Der größte Teil der Software kann sich auch auf dem Adapter befinden. Durch den Aufruf wird NetBIOS resident installiert und an den Interrupt 5CH angeschlossen, dessen direkter Aufruf einem BIOS-Ruf entspricht.
Die meisten Programme verwenden die Schnittstelle des BDOS-Interrupts 2AH. NetBIOS bietet – unabhängig vom zugrunde liegenden LAN – einen einheitlichen Netzzugang für Netzbetriebssysteme und verteilte Anwendungen. Es stellt einen Defacto-Standard dar, der für alle gebräuchlichen Netzadapter (sowohl auf Token-Ring- als auch auf Ethernet- und Arcnet-Basis) zur Verfügung steht. Die meisten DOS-basierten Betriebssysteme und viele verteilte Anwendungen setzen auf NetBIOS auf. Eine Ausnahme bildet Netware, das ein eigenes Protokoll – **IPX** – nutzt, aber trotzdem ein darauf basierendes NetBIOS für andere Programme enthält.
NetBIOS stellt gleichermaßen verbindungslose (im OSI-Referenzmodell in Ebene 2 einzuordnende) und verbindungsorientierte (sichere) Übertragungsdienste (Ebene 5) zur Verfügung. Die Identifikation der angesprochenen Stationen erfolgt über einen dezentralen, nichtredundanten Namensdienst, der eine flexible Internetzarbeit unterstützt, worunter in großen Netzen aber die Performance leidet. Somit steht der weiten Verbreitung und universellen Einsetzbarkeit ein ungünstiges Zeitverhalten gegenüber, das Novell bewog, ein eigenes Protokoll einzusetzen, während Microsoft beim LAN-Manager von OS/2 neben anderen Kommunikationsschnittstellen an NetBIOS festhält.
Das von Novell entwickelte Übertragungsprotokoll (IPX = Internetwork Packet Exchange) orientierte sich weitgehend am XNS-Protokoll (Xerox Network System) des Ethernet-Entwicklers Rank Xerox. Es handelt sich – wie am Namen zu erkennen – um ein

Internetprotokoll; die Adressen setzen sich aus einer Netz-, einer Knoten- und einer Prozeßnummer zusammen. Das Protokoll bietet verbindungslose Dienste, behandelt aber auch die Wegwahl, so daß es im OSI-Referenzmodell in Schicht 3 einzuordnen ist. Da IPX keine ausreichende Sicherheit bietet, wird zusätzlich das *Sequenced Packet Protocol* (SPX) zur Verfügung gestellt, das auf IPX aufsetzt und in der Funktionalität der OSI-Schicht 4 entspricht. Sie sehen, daß sich die einzelnen Entwicklungen zwar in das Referenzmodell einordnen lassen, eine vollständige Realisierung jedoch eine Idealvorstellung ist. In lokalen Netzen werden von den 7 geforderten Schnittstellen zumeist nur drei oder vier angeboten, was die Portierung einzelner Schichten erschwert, die Geschwindigkeit aber wesentlich erhöht.

Auf einen völlig anderen Ursprung als die beiden beschriebenen Protokolle geht **TCP/IP** (Transmission Control Protocol/Internet Protokoll) zurück, das insbesondere für die Verbindung unterschiedlicher Hardware und Betriebssysteme wachsende Bedeutung erlangt. In Ermangelung offizieller Standards gab das amerikanische Verteidigungsministerium (Department of Defense) Anfang der 70er Jahre die Entwicklung eines Kommunikationsprotokolles für den Datenaustausch in heterogenen Netzen in Auftrag, an dessen Entwicklung die University of California maßgeblich beteiligt war, wo zur gleichen Zeit das Betriebssystem Unix entstand. So ist es zu erklären, daß Berkley Unix und viele andere Implementierungen TCP/IP bereits enthalten. Der Name TCP/IP steht häufig auch für die darauf aufbauenden Applikationsprotokolle, zum Beispiel die Terminalemulation Telnet, die Dateitransferprotokolle FTP (File Transfer Protocol) und TFTP (Trivial FTP) oder das Mailsystem SMTP (Simple Mail Transfer Protocol), die eigentlich völlig selbständige Anwendungen darstellen. Interessant aus Sicht des PC-Nutzers ist, daß das Ethernet zur Anwendung kommt, so daß dieses Protokoll auf entsprechenden Controllerkarten problemlos eingesetzt werden kann und zumeist vom Hersteller auch angeboten wird. Spezielle Gatewaykarten wie die NP600G ermöglichen eine Protokollumwandlung on board. Die Verbindung von PC-Netzen mit Unix-Workstations oder VAX-Technik ist so möglich.

Netzbetriebssysteme

Netzbetriebssysteme stellen die wichtigste Softwarekomponente eines lokalen Netzes in der Anwenderebene dar. Sie ermöglichen die Ressourcen- und Datenteilung und stellen auch Schnittstellen für andere Netzwerkdienste zur Verfügung.

Das tragende Prinzip der zur Zeit eingesetzten PC-Netzbetriebssysteme, wie MS-NET, Novell NetWare oder der OS/2-LAN-Manager, ist die virtuelle Diskette. Bild 2 zeigt dieses Prinzip. Auf entfernte und lokale Ressourcen, zum Beispiel Dateien auf der Festplatte, wird in exakt derselben Weise zugegriffen, da die Anwendung trotz Netzwerkes nichts anderes als die übliche, etwas erweiterte Schnittstelle zum Betriebssystem benutzt. Ein als Erweiterung in das Betriebssystem eingebrachtes Modul erkennt, ob eine durch einen Systemaufruf angesprochene Ressource lokal oder entfernt residiert. Das in der Abbildung allgemein als Interzeptor (Abfänger = Abfangen der Systemdienstrufe) bezeichnete Modul ist der Redirector in MS-NET und OS/2 bzw. die Shell oder der Requestor in NetWare. Handelt es sich um eine entfernte Ressource, wird ein Dienstaufruf an den Server gesandt und dort bearbeitet.

Eine Datenbankabfrage in der in dBase üblichen Form

LIST FOR <bedingung>

würde in einem Netzbetriebssystem etwa so ablaufen: Zunächst wird ein OpenFile-Ruf an den Server gesandt, danach so lange ReadFile-Rufe, bis die entsprechenden Daten gefunden werden, worauf ein CloseFile-Ruf an den Server geht. Alle Daten werden im Netz übertragen, auch die, die nicht benötigt werden.

Es leuchtet ein, daß dieses Verfahren für Anwendungen, die über ein Arbeitsgruppennetz hinausgehende Dienste beanspruchen, wenig geeignet ist. Es setzt, um effektiv zu sein, eine weitestgehend homogene Umgebung sowohl in bezug auf die Diensterbringung (Protokoll an das PC-Betriebssystem angepaßt) als auch in bezug auf die hohe notwendige Übertragungsrate im Netz voraus. In eine unternehmensweite Kommunikationsinfrastruktur sollen aber gerade auch andere Systemarchitekturen integriert werden. Weiterhin kann nicht davon ausgegangen werden, daß alle Übertragungskanäle innerhalb des Gesamtsystems eine Übertragungskapazität besitzen, die das Prinzip der virtuellen Diskette geeignet unterstützen.

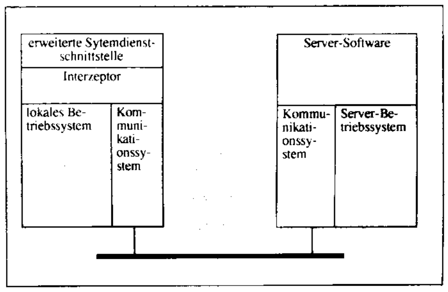


Bild 2 Prinzip eines Netzwerkbetriebssystems

Dies erkennend, geht man mit den in der Zwischenzeit verfügbaren technologischen Voraussetzungen dazu über, neben dem herkömmlichen Prinzip der virtuellen Diskette Mittel bereitzustellen, die netzwerkweite, verteilte Anwendungen unterstützen. Es wird nicht mehr nur ein Teil des Betriebssystems verteilt, sondern die Anwendung selbst. Be-

Tafel 3 Vergleich der wichtigsten Netzwerk-Betriebssysteme

	Advanced NetWare	NetWare 386	OS/2-LAN-Manager v2.0	VINES
Preis (DM)	6.500	16.000	durch OEM festgelegt	12.000
Anzahl der Nutzer für den Preis	100	unbegrenzt		unbegrenzt
Server-Operationen				
Festplattenformat	spezifisch	spezifisch	HPFS	Unix
Disk-Spiegelung	ja	ja	ja	nein
Anzahl LAN-Karten im Server	4	16	4	4
gleichzeitige Log-ins	100	250	1000	unbegrenzt
minimal empfohlener RAM	2 MByte	4 MByte	4 MByte	4 MByte
Workstation-Software				
max. verbleibender RAM, wenn DOS und Netzwerk geladen sind	540 KByte	556 KByte	525 KByte	600 KByte
dito mit NetBIOS geladen	516 KByte	516 KByte	525 KByte	553 KByte
kompatibel mit OS/2-Arbeitsstationen	ja	ja	ja	ja
Datei-System				
max. Anzahl gleichzeitig offener Dateien/Server	1.000	100.000	16.000	200.000
max. Anz. Volumes/Server	32	32	24	12
max. Anz. physischer Laufwerke/Volume	1	32	1	1
max. Anz. phys. Laufwerke pro Server	32	1.024	24	4
Datei kann über mehrere Laufwerke gehen	nein	ja	nein	nein
max. Dateigröße	255 MByte	4 GByte	2 GByte	unbegrenzt
max. Volume-Größe	255 MByte	32 TByte	2 GByte	unbegrenzt
max. adressierbarer RAM	12 MByte	4 GByte	16 MByte	24 MByte
max. Gesamt-Diskspeicher	32 GByte	32 TByte	48 GByte	2,4 GByte
Netzwerk-Verwaltung				
Fehler/Status Log	ja	ja	ja	ja
zeigt Anzahl fehlerhafter Pakete an	ja	ja	ja	ja
zeigt Netzwerkfehler an	ja	ja	ja	ja
zeigt offene Dateien an	ja	ja	ja	ja
zeigt Namen angemeldeter Benutzer an	ja	ja	ja	ja
zeigt aktuelle Server-Auslastung an	ja	ja	nein	ja
Sicherheit				
Zugriff auf Netzwerkressourcen kann Gruppen zugeordnet werden	ja	ja	ja	ja
Zugriff kann zeitlich eingeschränkt werden	ja	ja	ja	ja
Paßwörter können mit Netzressourcen verbunden werden	ja	ja	ja	ja
verschlüsselte Paßwörter können über das Netz gesendet werden	nein	ja	ja	ja
elektronische Post				
Post kann zwischengespeichert werden	nein	nein	nein	ja
Binär-Dateien können mit der Post gesandt werden	nein	nein	nein	ja
System erlaubt Kurznachrichten	ja	ja	nein	ja
Druckdienst				
Druck-Spooler verfügbar	ja	ja	ja	ja
Nutzer kann Druck-Warteschlange modifizieren	ja	ja	ja	ja
Anzahl verwalteter Drucker	5	16	unbegrenzt	unbegrenzt
Drucker an Arbeitsstationen werden unterstützt	nein	ja	ja	ja
Drucker kann mit Softfonds betrieben werden	ja	ja	ja	ja

zogen auf das obige dBase-Beispiel bedeutet das, daß eine Dienstanforderung **LIST FOR bedingung** an einen speziellen Datenbankserver übergeben wird. Der Server bearbeitet die Anforderung lokal und sendet nur die tatsächlich angeforderten Daten (also das Ergebnis) an den Auftraggeber (Client) zurück. Das Dienstprotokoll ist nicht mehr betriebssystemspezifisch, sondern anwendungsspezifisch, und der Kommunikationsaufwand wird geringer. Besonders interessant wird diese Konfiguration, wenn der Server eine Standardschnittstelle anbietet, wie beispielsweise bei einem Datenbankserver eine SQL-Schnittstelle. In diesem Falle lassen sich um den Server eine Vielzahl von Standardanwendungen gruppieren, zum Beispiel Tabellenkalkulation, Finanzbuchhaltung usw.

Drei große Netzwerkbetriebssysteme werben heute um die Gunst der Endnutzer: Novells NetWare, Microsofts OS/2-LAN-Manager in seinen verschiedenen OEM-Versionen und Banyans VINES (siehe Tafel 3). Die drei Systeme sollen überblicksartig kurz vorgestellt werden.

NetWare

von Novell zählt zu den besten und leistungsfähigsten Betriebssystemen für PC-LANs. Bis zum Erscheinen des OS/2-LAN-Managers hatte es auf dem europäischen Markt keine wesentlichen Mitbewerber. In den USA stand ihm bis dahin nur Banyans VINES nennenswert gegenüber. Seit seiner Markteinführung 1983 wurde es ständig den neuesten Entwicklungen auf dem Hard- und Softwaremarkt angepaßt. Die aktuelle Version ist NetWare 386 3.0. Seit August wird die Version 3.1 angeboten.

Der Grundaufbau entspricht dem in Bild 2. Auf seiten der Arbeitsstation wird das Betriebssystem im Falle von MS-DOS um die NetWare Shell erweitert. Das Übertragungssystem wird durch das IPX-Protokoll gebildet. Serverseitig setzt das Betriebssystem direkt auf die Hardware auf. Diese Architektur ist wesentlich für die Leistungsfähigkeit von NetWare verantwortlich.

Die Verbindung zwischen Shell und Datei-Server erfolgt direkt unter Umgebung des Betriebssystems auf der Arbeitsstation. Im *NetWare File-Service Core Protocol* (NCP) sind 150 Dienstleistungsaufrufe vereinigt, mit deren Hilfe über die Shell der lokalen Anwendung die Netzwerk-Dienste bereitgestellt werden können.

Neben der Shell als Schnittstelle für netzwerksfähige Anwendungen können verteilte Client-Server-Anwendungen direkt das Übertragungssystem nutzen. Bei NetWare 286 standen als Protokolle dafür bislang nur das IPX, SPX und NetBIOS sowie durch andere Anbieter auch TCP/IP zur Verfügung. Mit der neuen Architektur von NetWare 386 ist prinzipiell die Nutzung beliebiger Übertragungsprotokolle möglich. Dies wird einerseits durch das *Open Data Link Interface* (ODI) – einer Schnittstellenspezifikation zum LAN-Adapter – sowie andererseits durch die NetWare Streams erreicht, die einen einheitlichen Zugang vom Datei-Server-Protokoll bzw. Anwendungsprotokoll zum Übertragungssystem bilden.

Wesentliche Kriterien für die Bewertung von Netzbetriebssystemen sind deren Leistungsfähigkeit, Zuverlässigkeit und Möglichkeiten zum Daten- und Zugriffsschutz.

Zur Steigerung der Leistungsfähigkeit wurden ins NetWare-Datei-Server-Betriebssystem eine Reihe von Algorithmen aufgenommen. Dazu gehören:

– *Directory Caching:* Inhaltsverzeichnisse der Festplatte werden im RAM gehalten.
– *Directory Hashing:* Inhaltsverzeichnis wird für schnelle Suche aufbereitet.
– *File Caching:* Häufig benutzte Dateien werden im RAM gehalten.
– *Elevator Seeking:* Zugriffsaufträge werden nach Lage auf der Platte abgearbeitet.
– *Turbo-Fat:* Die File Allocation Table kann auf große Dateien indiziert werden.
– *verteilte Intelligenz durch Disk-Koprozessor.*

Die Zuverlässigkeit des Systems wird durch folgende Eigenschaften verbessert:

– Unterbrechungsfreie Stromversorgung (DSU) und USV-Monitoring
– redundante Verzeichnisstrukturen
– Read after Write Verification
– gespiegelte Laufwerke
– gedoppelte Laufwerke
– Transaktionsverfolgung
– dynamische Aussonderung defekter Plattenbereiche.

Schließlich verfügt NetWare über ein ausgetüfteltes, mehrstufiges Zugriffsschutz-System auf die verwalteten Datenbestände. Damit ist es möglich, jedem Benutzer individuell die für ihn notwendige und optimale Betriebsumgebung zu schaffen. Die einzelnen Stufen des Sicherheitssystems umfassen:

– *Login- und Passwort-Schutz:* Zugang zum Netz nur durch eingetragene Benutzer möglich
– *Trustee-Schutz:* An jeden Nutzer werden konkrete Nutzungsrechte, bezogen auf die einzelnen von ihm benutzbaren Verzeichnisse, vergeben.
– *Verzeichnis-Schutz:* Unabhängig von den Nutzerrechten, bezogen auf ein Verzeichnis, werden für das Verzeichnis selbst noch einmal Nutzungsrechte vergeben.
– *Datei-Attribute-Schutz:* Über Datei-Attribute kann für jede Datei festgelegt werden, wie mit ihr gearbeitet werden darf.

Aus der Kombination der verschiedenen Rechte ergeben sich die effektiven Rechte des Nutzers, bezogen auf die vom Betriebssystem verwalteten Objekte.

Im Umfeld von NetWare existiert eine Vielzahl von Produkten zur Integration mit anderen Systemarchitekturen (Host-Anbindung, Übergang von LANs zu WANs usw.). Um den eingangs erwähnten Gedanken der Verbindung von MS-DOS und Unix aufzugreifen, so bieten sich insbesondere im Zusammenhang mit Netware einige interessante Wege in die Zukunft: Unter dem Namen Portable Netware stellt Novell für Systemanbieter Quellen zur Implementation eines Netware-Servers unter Unix oder auch anderen Betriebssystemen (z. B. VMS) zur Verfügung. Ein anderer Weg wird mit dem Austausch der Protokolle bei NetWare 386 beschritten. Statt des Hausprotokolls können die Server so mit TCP/IP arbeiten – dem bevorzugten Unix-Protokoll. Da von beiden Systemen das Ethernet verwendet wird, ist die technische Anbindung kein Problem.

LAN-Manager

von Microsoft ist die Weiterentwicklung des MS-NET für das Betriebssystem OS/2. Während das Einplatzsystem MS-DOS zur Netzarbeit überlistet werden mußte, bietet das Multitasking-System OS/2 bessere Voraussetzungen für den Netzbetrieb. Dadurch konnte der LAN-Manager zum echten Konkurrenten von NetWare werden.

Im Gegensatz zu NetWare ist der Server eine Erweiterung von OS/2. Der Interzeptor-Modul auf der Arbeitsstation heißt in diesem Falle Redirector. Er existiert für MS-DOS, Windows und OS/2.

Eine wesentliche Eigenschaft des LAN-Managers ist die sowohl vertikale wie horizontale Kompatibilität. Vertikale Kompatibilität bedeutet in diesem Zusammenhang, daß alle drei Standardbetriebssysteme im Netz arbeiten können. MS-DOS, OS/2 und Unix. Horizontale Kompatibilität bedeutet, daß die verschiedenen Hersteller die Möglichkeit erhalten, eigene, individuelle Problemlösungen auf der Basis gemeinsamer Normen zu schaffen. Es gibt derzeit Versionen von über 50 Lizenznehmern, die alle untereinander kompatibel sind, jedoch auch herstellereigene Besonderheiten aufweisen. So hat zum Beispiel die IBM-Version spezielle Merkmale zur IBM-Host-Integration, während die 3Com-Version 3+Open ein spezielles Mail-System besitzt.

Die bisher verfügbare Version 1.1 des LAN-Managers konnte sich nicht als wirklicher Konkurrent zu NetWare etablieren. Das lag an dem unzureichenden Datenschutz und der zu geringen Leistungsfähigkeit. Mit der Version 2.0, die zur Zeit an die OEM-Partner von Microsoft ausgeliefert wird, werden diese Nachteile beseitigt. Wesentliche Neuerung, die für eine Leistungssteigerung verantwortlich zeichnet, ist das *High Performance File System* (HPFS), das nun anstelle der *File Allocation Tables* (FAT) benutzt wird. Erhöhte Zuverlässigkeit wird durch eingebaute Fehlertoleranz, ähnlich wie bei NetWare, erreicht. Auch der Zugriffsschutz bietet ein ähnliches Niveau, wie bei NetWare.

Durch die Anwendung verschiedener Industriestandard-Protokolle kann der LAN-Manager auf einer Vielzahl von Netzwerken laufen. Eine dem *NetWare ODI* entsprechende Schnittstelle zwischen Netzbetriebssystem und LAN-Adapter ist die *Network Driver Interface Specification* (NDIS). Jede Adapterkarte, die diese Schnittstelle anbietet, ist also unter dem LAN-Manager einsetzbar. Die Architektur des LAN-Managers erlaubt den Einsatz von IBMs *NetBIOS Extended User Interface* (NetBEUI), OSI-TP4, TCP/IP und XNS. Ein leistungsfähiger Mechanismus zur Interprozeßkommunikation von Client-Server-Anwendungen sind die *Named Pipes*, eine NetBIOS ähnliche Schnittstelle, die aber leistungsfähiger und leichter zu benutzen ist. Als Datei-Server-Protokoll werden wie schon in MS-NET, aber mit erweitertem Funktionsumfang, die *Server Messages Blocks* (SMBs) eingesetzt.

VINES

steht für ***VIrtuel NEtwork System***. Aufgrund der bisherigen Vermarktungsstrategie des Herstellers Banyan hat es in Deutschland bisher wenig Verbreitung gefunden. In den USA dagegen ist es recht weit verbreitet.

Die Server-Software basiert auf Unix System V. Als reines PC-Netzwerk besitzt es ähnliche Eigenschaften wie NetWare oder der LAN-Manager. Seine wirklichen Stärken zeigt VINES vor allem in einem Internetz mit mehreren Severn und Teilnetzen im LAN- und WAN-Bereich.

Dies wird durch einen in VINES integrierten Modul – StreetTalk – erreicht. StreetTalk ist ein leistungsfähiges, virtuelles Verzeichnissystem. Es unterhält eine verteilte Datenbank, die den Namen, den Ort und die Attri-

bute eines jeden Objektes im Netzwerk aufführt. Dieses globale, verteilte Verzeichnissystem stellt den Mittelpunkt der VINES-Architektur dar.
Der Benutzer braucht sich nicht um Netzwerktopologien, Kommunikationsprotokolle oder den Ort der benutzten Objekte zu kümmern. Hat er die Berechtigung zum Zugriff auf ein Objekt, muß er nur die im gesamten Netz einmalig vorkommende Benennung für dieses Objekt angeben, um den Zugriff zu erhalten. In anderen Systemen muß man sich explizit bei dem Server anmelden, auf dem das Objekt residiert.

Was noch zu beachten ist

Lokale Netze erreichen heute eine ausreichende Geschwindigkeit, sind technisch ausgereift und werden softwaremäßig breit unterstützt, so daß sie auch als Lösung für anspruchsvolle Aufgabenstellungen in Betracht gezogen werden können. Die auftretenden Probleme sollen aber nicht verschwiegen werden: Mit bedingungsloser Kompatibilität wie bei MS-DOS darf man bei großen Systemen (OS/2, Unix, Netware) nicht mehr rechnen. Größtes Handicap sind die verschiedenen Festplattencontroller, mit denen die Software nicht zusammenspielt. So mußten wir zur Installation sowohl von Netware 386 als auch von SCO Unix 386 eine andere Festplatte (SCSI) einschließlich Controller beschaffen, da die Originalteile des Rechnerherstellers keines der beiden Systeme unterstützten. In kleinerem Umfange kann es Probleme auch mit anderen Karten geben (bei Netzwerkkarten ist besonders auf das Vorhandensein der richtigen Treiber zu achten). Um sicher zu gehen, daß die Betriebssysteme laufen, muß entweder einer der Referenzrechner eingesetzt werden (die Softwarehersteller bieten zumeist eine Liste von Geräten, auf denen die Lauffähigkeit getestet wurde), oder man überläßt die gesamte Planung einem Spezialisten, der über entsprechende Erfahrung verfügt und bereits einige Installationen hinter sich hat. Ähnliches gilt auch für die Software, wo sehr wohl zu unterscheiden ist zwischen einer echt verteilten Anwendung, netzfähiger oder nur netzverträglicher Software. Deshalb kommt der Planung und Installation eines Netzes besondere Bedeutung zu, die später viel Arbeit und Geld sparen kann. Gleiches gilt auch für die Festlegung von Zugriffsrechten und Maßnahmen zur Datensicherung.

✉ KONTAKT ☎

Gecko – Gesellschaft für Computer und Kommunikationssysteme mbH, F.-Stillfried-Straße 4, 2500 Rostock 1

Die wichtigsten Netzwerk-Fragen

Starthilfe

Werner Pliegl

Obwohl Netzwerke momentan zu den beherrschenden Computerthemen gehören, sind für viele Interessierte noch eine Menge fundamentaler Fragen offen. Computer Persönlich forschte deshalb einmal in der Hotline der Firma Dell nach. Dort führt man genau Statistik über die verschiedenen Probleme der Netzwerk-User. Dabei wurden die Spitzenreiter der Fragen-Hitparade herausgefischt. Hier sind Sie:

▶ Was ist der Unterschied zwischen einem dedizierten und einem nichtdedizierten Fileserver?
Ein dedizierter Server übernimmt nur Aufgaben der Netzwerksteuerung. Ein nichtdedizierter Server kann dagegen gleichzeitig noch als Arbeitsplatz mitverwendet werden. Der Computer muß seine Kapazität also sowohl dem Netz als auch dem Arbeitsplatz zur Verfügung stellen. Die Performance im Netz sinkt dadurch. Ein weiterer Nachteil ist die geringere Datensicherheit. Der Vorteil der nichtdedizierten Lösung sind die geringeren Hardware-Kosten. Daher kommt das Prinzip für preiswerte oder Einsteigerlösungen in Betracht.

▶ Welche Computer kann man in ein vorhandenes Netz integrieren?
Das hängt vom Netzwerk-Betriebssystem ab. Prinzipiell lassen sich alle MS-DOS-kompatiblen PCs einbinden, sofern sie einen freien Steckplatz für die Netzwerkkarte besitzen. Bei Novell Netware lassen sich zum Beispiel PCs mit AT-Bus, MCA und EISA und Apple Macintosh einbinden. Bei manchen Netzwerksystemen ist auch eine Verbindung zu Minirechnern oder Unix-Anlagen möglich.

▶ Wo kann ein Bandlaufwerk zur Datensicherung eingebaut werden?
Der Platz für den Streamer hängt vom Betriebssystem ab. Bei Novell Netware 286 müssen sich Streamer in den Arbeitsstationen befinden. Netware 386 erlaubt den Einbau des Steamers in den Server.

▶ Wo muß ein Netzwerkdrucker angeschlossen werden?
Netware-Versionen bis 2.15 lassen den Anschluß von Netzwerkdruckern nur am Server zu. Dieses Manko läßt sich aber mit Utilities wie Printer Assist umgehen. Netware 386 ist von Haus aus flexibler: Netzwerkdrucker dürfen an jede Arbeitsstation angeschlossen werden und können von jedem Arbeitsplatz aus angesprochen werden.

▶ Welche Software läuft im Netz?
Im Grunde läßt sich jede Software auf dem Server installieren. Das Betriebssystem sichert geöffnete Dateien gegen Änderungen von anderen Arbeitsstationen. Vor dem Einsatz muß jedoch geklärt werden, ob die Lizenz sich auf diesen Einsatz erstreckt. Optimal sind dagegen spezielle Netzwerkversionen, die es von vielen Anwendungen gibt. Mit ihnen können mehrere Anwender gleichzeitig mit den gleichen Daten arbeiten.

▶ Wie groß sollte die Festplatte des Servers in etwa sein?
Richtschnur für eine vernünftige Auslegung ist die addierte Festplattenkapazität aller Arbeitsstationen im Netz. So groß sollte in etwa die Platte des Servers sein. Der Bedarf verdoppelt sich, wenn man auf erhöhte Datensicherheit Wert legt. Dazu braucht man zwei gleiche Festplatten, auf denen alle Daten doppelt gehalten werden (Disk Mirroring).

▶ Welche Netzwerk-Topologien gibt es?
Grundsätzlich gibt es drei Netzwerk-Topologien, Bus, Stern und Ring. Am häufigsten eingesetzt ist wohl die Bus-Topologie. Dabei hängen alle Arbeitsplätze hintereinander an derselben Leitung. Das Busprinzip wird bei Ethernet- oder Cheapernet-Verkabelung angewendet. Es läßt hohe Übertragungsraten zu. Bei hoher Auslastung steigt jedoch die Häufigkeit von Datenkollisionen, was sinkende Geschwindigkeit bewirkt. Bei der Sterntopologie führt von einem zentralen Verteiler eine eigene Leitung zu jeder Station. Dadurch sinkt die Gefahr von Datenkollisionen. Ein Beispiel für ein sternförmiges Netz ist Arcnet. Beim Token-Ring sind die einzelnen Stationen ringförmig miteinander verbunden. Hier sind Datenkollisionen ausgeschlossen. Jede Station erhält nämlich nur dann Senderecht, wenn sie im Besitz des Tokens (eines ständig zirkulierenden Datenpaketes) ist.

▶ Was ist ein Boot-PROM?
Einen Boot-PROM benötigt man beim Einsatz von Diskless-Workstations. Diskless-Workstations sind Netzwerk-PCs ohne Disketten- und Festplatten-Laufwerke. Sie bieten besseren Datenschutz (kein Kopieren möglich) und geringe Kosten. Da sich auf ihnen kein Betriebssystem befindet, müssen sie mit dem Betriebssystem des Server geboten werden. Der Boot-PROM auf dem Netzwerkadapter leitet diesen Vorgang ein.

▶ Wie werden neue Teilnehmer und Paßwörter eingerichtet?
Für die Zulassung neuer Benutzer ist der Netzwerkadministrator zuständig. Dazu bieten die meisten Netzwerksysteme menügesteuerte Utilities an. Mit ihm teilt man Zugriffsrechte, Paßwörter und Benutzernamen zu.

Nachdruck aus Computer Persönlich, Heft 14/1990

Netzwerk-Vorstellung

Dem Thema Netzwerke waren zwei Präsentationstage in Chemnitz gewidmet, mit dem das Datenverarbeitungszentrum, das zukünftig unter dem Namen COMPLEX Informationsverarbeitung GmbH firmiert, und MULTiTEC Computer Systeme GmbH mit Sitz in Chemnitz – Mitglied einer bundesdeutschen DV-Unternehmensgruppe, die sich schwerpunktmäßig mit Netzwerken befaßt – einem ausgewählten Anwenderkreis die Möglichkeit der Rechnerkopplung aufzeigte.
Vorgestellt wurden zwei Netzwerke auf Basis Novell, ausgestattet mit Netzwerkkarten von Schneider & Koch (SK), die PCs ganz unterschiedlicher Hersteller zusammenführten; unter anderem wurde auch ein EC 1834 in einem Netzwerk als Arbeitsplatz installiert. Diese Einbindung bedeutet, daß im PC-Bereich bis heute getätigte Investitionen erhalten bleiben und bei der Gestaltung von PC-Netzwerken als Arbeitsplätze mit berücksichtigt werden können.
Heterogene LANs sind Netzwerke, an die Rechner verschiedener Bauart angeschlossen werden können. So werden künftig historisch gewachsene Insellösungen zu einem Netz verbunden, und jeder kann auf die Daten der im Netz integrierten Rechner zugreifen, wenn er die entsprechenden Zugriffsrechte besitzt. Das bedeutet, daß von dem Arbeitsplatz über einen PC auf die Daten im Host-Rechner, im Unix-Rechner und auf dem Netzwerk-Server zugegriffen werden kann und teure Peripheriegeräte nur einmal angeschafft werden müssen, denn alle Netzteilnehmer nutzen beispielsweise Matrix- oder Laserdrucker oder Plotter gemeinsam. Realisiert werden können solche komplexen Netzwerke u. a. mit dem Netzwerkbetriebssystem Novell und den Netzwerk-Komponenten von SK. Diese Karten unterstützen vier verschiedene Protokolle und nutzen die Übertragungsmöglichkeiten des Ethernet/Cheapernet-Kabels voll aus. *MP*

Einer für (fast) alle
Screen-Safer

Christian Hanisch, Berlin

Oft bleibt der eingeschaltete PC für längere Zeit in untätiger Bereitschaftsstellung. In solchen Arbeitspausen ist es sinnvoll, wenn der Bildschirm dunkel geschaltet wird, um die Bildröhre vor einem Einbrennen der Anzeige zu bewahren.

Unter der Bezeichnung Screen-Safer, Nightprogram, Blackmon und so weiter sind Programme bekannt geworden, die den Bildschirm nach einer gewissen Zeitspanne der Untätigkeit dunkel schalten und ihn beim nächsten Tastendruck wieder aktivieren. Es handelt sich hier ausnahmslos um residente Programme, die den Timer-Interrupt und den Tastatur-Interrupt überwachen. In der Regel wird ein Timer-Interrupt-Handler verwendet, der einen Zähler beständig dekrementiert. Diesem Prozeß wirkt der Tastatur-Interrupt-Handler entgegen, indem er bei jedem Tastendruck den Zähler auf seinen Ausgangswert zurücksetzt. Dieser Ausgangswert kann im Programm fest vorgegeben sein oder beim Programmstart interaktiv festgelegt werden. Er steht für die Zeitspanne, die ohne eine Tastenbetätigung verstreichen muß, ehe der Bildschirm – meist beim Zählerstand Null – dunkel geschaltet wird. Beim nächsten Tastendruck sorgt der Tastatur-Interrupt-Handler bei dunklem Bildschirm dann auch dafür, daß der Bildschirm wieder angeschaltet wird.

Nun sind eine Reihe verschiedener Grafikkarten in Gebrauch, die zum Teil unterschiedliche Mechanismen für das An- und Abschalten des Bildschirmes erfordern. In der Literatur wurden bisher häufig nur Programme für ganz bestimmte Karten und ausschließlich nach der oben beschriebenen Arbeitsweise vorgestellt. Hier möchte ich Ihnen nun ein Programm anbieten, das nach automatischer Erkennung der Grafikkarten für die Hercules-, MDA-, CGA-, EGA- und VGA-Karte unter allen Grafikmodi arbeitet. Darüber hinaus bietet das Programm einen weiteren Service, indem ein Interrupt-10H-Handler einbezogen wird. Dessen Aufgabe kann analog zu der des Tastatur-Interrupt-Handlers gesehen werden. Bei einer Ausgabe auf den Bildschirm via INT 10H schaltet er denselben gegebenenfalls an, beziehungsweise bei beständiger Ausgabe verhindert er das Abschalten des Bildschirms. Mit dieser Option, die wahlweise benutzt werden kann, wird beabsichtigt, das Abschalten des Bildschirms bei solchen Programmen zu unterbinden, die rechenintensiv sind und während der Arbeit keine Eingabe erfordern, wohl aber für regelmäßige Bildschirmausgaben sorgen.

Das Programm ADS arbeitet mit Kommandozeilenparametern. Als erster Parameter wird die Zeit in Sekunden angegeben, die bis zur Dunkelschaltung verstreichen soll, wenn keine Taste gedrückt wird beziehungsweise keine Bildschirmausgabe von Programmen aus erfolgt. Letztere Bedingung steht für den Interrupt-10H-Service und kann mit einem zweiten Aufrufparameter verhindert werden, wodurch das Programm in traditioneller Weise funktioniert. Die Uhr des Norton-Commanders wird beispielsweise über Interrupt 10H bedient. Ist sie eingeschaltet und ADS mit Interrupt-10H-Service aktiv, so verhindert sie die Dunkelschaltung des Bildschirms. ADS müßte in diesem Falle mit einem zweiten Kommandozeilenparameter ('X') aufgerufen werden:

C:\>ADS 180 X<CR>

Das Programm gliedert sich in einen transienten und in einen residenten Teil. Der transiente Teil beinhaltet die Initialisierung der Variablen, das Feststellen der Grafik-Hardware, das Testen auf einer bereits bestehenden Installation von ADS sowie das Installieren der Interrupt-Handler. Der residente Teil besteht aus insgesamt vier Interrupt-handlern.

Der Timer-Interrupt-Handler

Dieser Interrupt-Handler wird in den Interrupt 1CH eingefügt und kontrolliert den Zählerstand der Variablen *COUNTER*. Ist dieser Null, so wird der Bildschirm durch den Aufruf der Prozedur *SWITCH_BOARDS* dunkel geschaltet, und das Flag *SWITCH* auf *off* gesetzt. Bei der Prozedur *SWITCH_BOARDS* wurde vom Konzept der prozeduralen Parameter in Turbo Pascal 5.x Gebrauch gemacht. *SWITCH_BOARDS* wird bei der Initialisierung die für die entsprechende Hardware relevante Routine zugewiesen. Damit ist programmtechnisch der universelle Anschluß beliebiger Routinen zur Laufzeit realisiert.

Der Timer-Interrupt-Handler sorgt bei einem Zählerstand ungleich Null dafür, daß die Variable *COUNTER* dekrementiert wird. Der Aufruf des alten Interrupt 1CH ist notwendig,

Bild 1 Programm ADS.PAS

```
  1: (*$M 1024,0,0*)
  2: (*$N-,S-,F-*)
  3: PROGRAM Auto_Display_Switch; (* Turbo Pascal 5.x *)
  4: USES crt,dos;
  5: CONST ver   = '2.6';
  6: TYPE flipflop = (off,on);
  7:      hardware = PROCEDURE (y:flipflop);
  8:      boards   = (hrc, mda, cga, ega, vga, unknown);
  9:
 10: VAR  counter, initvalue, code,min, sec : INTEGER;
 11:      oldvec9, oldvec1c,
 12:      oldvec10, oldvec11, vektor       : pointer;
 13:      switch_boards                    : hardware;
 14:      equip                            : word ABSOLUTE $40:$10;
 15:      initvaluestr                     : STRING[8];
 16:      switch                           : flipflop;
 17:      selector                         : boards;
 18:      c                                : CHAR;
 19:      int10                            : BOOLEAN;
 20:      regs                             : registers;
 21:
 22: PROCEDURE call_int(vektor:pointer);
 23: INLINE($8f/$06/vektor/        { hole Adresse vom Stack      }
 24:        $8f/$06/vektor+2/      {            "                }
 25:        $9c/                   { Pushf                       }
 26:        $fa/                   { cli                         }
 27:        $ff/$1e/vektor);       { call far [vektor]           }
 28:
 29: (*$F+*)
 30: PROCEDURE switch_cga(y:flipflop);
 31: VAR status : BYTE ABSOLUTE $40:$65;
 32: BEGIN
 33:   IF y=off THEN port[$3d8]:=status AND $f7
 34:            ELSE port[$3d8]:=status OR  $08;
 35: END;
 36:
 37: PROCEDURE switch_mda(y:flipflop);
 38: VAR status : BYTE ABSOLUTE $40:$65;
 39: BEGIN
 40:   IF y=off THEN port[$3b8]:=status AND $f7
 41:            ELSE port[$3b8]:=status OR  $08;
 42: END;
 43:
 44: PROCEDURE switch_ega(y:flipflop);
 45: VAR d:BYTE;
 46: BEGIN
 47:   d:=port[$3ba];
 48:   d:=port[$3da];
 49:   IF y=off THEN port[$3c0]:=$0
 50:            ELSE port[$3c0]:=$20;
 51: END;
 52:
 53: PROCEDURE switch_8514(y:flipflop);
 54: BEGIN
 55:   IF y=off THEN regs.ax:=$1201
 56:            ELSE regs.ax:=$1200;
 57:   regs.bl:=$36;
 58:   intr($10,regs);
 59: END;
 60:
 61: PROCEDURE switch_hgc(y:flipflop);
 62: VAR i:word; x:BYTE;
 63: BEGIN
 64:   IF y=off THEN BEGIN
 65:     port[$3b4]:=6;
 66:     port[$3b5]:=0;
 67:   END ELSE INLINE(
 68:     $31/$ff           {          xor  di,di                          }
 69:    /$B9/$03/$00       {          mov  cx,0003h ;3 Druchläufe          }
 70:    /$30/$C0           { e1:      xor  al,al    ;das muß reichen!      }
 71:    /$BA/$BA/$03       {          mov  dx,03ba                         }
 72:    /$EC               { v_sync1:in   al,dx                            }
 73:    /$A8/$80           {          test al,80h   ;auf Begin von         }
 74:    /$74/$FB           {          jz   v_sync1  ;Retrace warten        }
 75:    /$EC               { v_sync2:in   al,dx                            }
 76:    /$A8/$80           {          test al,80h   ;auf Ende von          }
 77:    /$75/$FB           {          jnz  v_sync2  ;Retrace warten        }
 78:    /$30/$C0           {          xor  al,al    ;AL=0                  }
 79:    /$BA/$BB/$03       {          mov  dx,03bbh ;Light-Pen-Register    }
 80:    /$EE               {          out  dx,al    ;löschen               }
 81:    /$BA/$B9/$03       {          mov  dx,03b9h                        }
 82:    /$EE               {          out  dx,al    ;Light-Pen setzen      }
 83:    /$B0/$10           {          mov  al,10h                          }
 84:    /$BA/$B4/$03       {          mov  dx,03b4h ;Index-Register        }
 85:    /$EE               {          out  dx,al    ;mit 10h laden         }
 86:    /$42               {          inc  dx                              }
 87:    /$EC               {          in   al,dx    ;Light-Pen-Adresse     }
 88:    /$88/$C7           {          mov  bh,al    ;High-Teil nach BH     }
 89:    /$B0/$11           {          mov  al,11h                          }
 90:    /$BA/$B4/$03       {          mov  dx,03b4h ;Index-Register        }
 91:    /$EE               {          out  dx,al    ;mit 11h laden         }
 92:    /$42               {          inc  dx                              }
 93:    /$EC               {          in   al,dx    ;Light-Pen-Adresse     }
 94:    /$88/$C3           {          mov  bl,al    ;Low-Teil nach BL      }
 95:    /$39/$FB           {          cmp  bx,di                           }
 96:    /$72/$02           {          jb   skip                            }
 97:    /$89/$DF           {          mov  di,bx                           }
 98:    /$E2/$CB           { skip:    loop e1                              }
 99:    /$4A               {          dec  dx                              }
100:    /$B0/$06           {          mov  al,06h   ;Index-Register auf    }
101:    /$EE               {          out  dx,al    ;Zeilenzahl            }
102:    /$42               {          inc  dx                              }
103:    /$B0/$19           {          mov  al,19h   ;25 Zeilen für Text    }
104:    /$81/$FF/$00/$09   {          cmp  di,0900h ;Schwellenwert?        }
105:    /$72/$02           {          jb   text     ;darunter? dann Text   }
106:    /$B0/$57           {          mov  al,57h   ;87 Zeilen für Grafik  }
107:    /$EE);             { text:    out  dx,al    ;Zeilenzahl senden     }
108: END;
109: (*$F-*)
110:
```

```
111: PROCEDURE own_1C; INTERRUPT;
112: BEGIN
113:   IF counter=0 THEN BEGIN
114:     switch_boards(off); switch:=off
115:   END ELSE dec(counter,1);
116:   call_int(oldvec1c);
117: END;
118:
119: PROCEDURE own_9; INTERRUPT;
120: BEGIN
121:   IF switch=off THEN BEGIN
122:     switch_boards(on); switch:=on
123:   END;
124:   counter:=initvalue;
125:   call_int(oldvec9);
126: END;
127:
128: FUNCTION hercules:BOOLEAN; EXTERNAL;
129: FUNCTION lightpen:BOOLEAN; EXTERNAL;
130: PROCEDURE own_10; EXTERNAL;
131: PROCEDURE own_11; EXTERNAL;
132: (*$L ext.obj*)
133:
134: PROCEDURE terminate(message:STRING);
135: VAR attr:BYTE;
136: BEGIN
137:   attr:=textattr;
138:   textcolor(13);
139:   writeln(message);
140:   writeln('**** Cancel installation!');
141:   textattr:=attr;  halt(255)
142: END;
143:
144: BEGIN
145:   writeln; highvideo;
146:   writeln('Auto_Display_Switch for '
147:     'Hercules, MDA, CGA, EGA, VGA (Version ',ver,')');
148:   writeln('(c) Ch. Hanisch, 1989'); writeln; lowvideo;
149:   IF paramcount=0 THEN terminate('Parameter missing ... !'+
150:     #13#10+'Call: ADS <time to switch off> [<''X'' to '+
151:     'disable INT 10h Service>]')
152:                   ELSE BEGIN
153:     initvaluestr:=paramstr(1);
154:     val(initvaluestr,initvalue,code);
155:     IF code<>0 THEN terminate('Bad parameter!')
156:               ELSE initvalue:=round(initvalue*18.2);
157:     fillchar(regs,sizeof(regs),#0);
158:     intr($11,regs);
159:     IF regs.bx=$AAAA THEN terminate('ADS already installed!');
160:   END;
161:   textcolor(white);
162: (* Ermittlung der Grafik-Hardware *)
163:   selector:=unknown;
164:   IF (equip AND $0030=$0030) THEN BEGIN (* monochrom *)
165:     selector:=mda;
166:     IF hercules THEN BEGIN
167:       selector:=hrc;
168:       IF NOT lightpen THEN BEGIN
169:         textattr:=textattr+blink;
170:         writeln('There could be same problems!':54);
171:         writeln('The Hercules-Board has no light-pen.':58);
172:         writeln;
173:         textattr:=textattr AND (NOT blink);
174:         write('Cancel installation? (Y/n) ');
175:         c:=upcase(readkey);
176:         IF c='N' THEN writeln('N')
177:                  ELSE BEGIN writeln('Y');
178:                         terminate('User break')
179:                       END;
180:       END
181:     END
182:   END ELSE BEGIN (* color *)
183:     IF mem[$0040:$00A8]=0 THEN selector:=cga ELSE BEGIN
184:       regs.ax:=$1200; regs.bx:=$0010;
185:       intr($10,regs);
186:       IF regs.bl<>$10 THEN selector:=ega;
187: (*    regs.ah:=$1a; intr($10,regs);
188:       if regs.al=$1a then selector:=vga;                    *)
189:     END
190:   END;
191:
192: (* Switch-Boards bekommt die richtige Routine zugewiesen   *)
193:   CASE selector OF
194:     hrc: BEGIN
195:            switch_boards:=switch_hgc; write('TTL Monochrom');
196:          END;
197:     mda: BEGIN
198:            switch_boards:=switch_mda; write('MDA');
199:          END;
200:     cga: BEGIN
201:            switch_boards:=switch_cga; write('Color');
202:          END;
203:     ega: BEGIN
204:            switch_boards:=switch_ega; write('EGA / VGA');
205:          END;
206:     vga: BEGIN
207:            switch_boards:=switch_8514; write('VGA / 8514');
208:          END;
209:     ELSE terminate('Unknown video board!');
210:   END;
211:   writeln(' board active ...');
212:   counter:=initvalue;
213:
214: (* Installation der Interrupt-Handler *)
215:   swapvectors;
216:   getintvec($1c,oldvec1c);
217:   setintvec($1c,@own_1c);
218:   getintvec($9,oldvec9);
219:   setintvec($9,@own_9);
220:   IF paramcount=2 THEN int10:=FALSE ELSE BEGIN
221:     getintvec($10,oldvec10);
222:     setintvec($10,@own_10);
223:     int10:=TRUE;
224:   END;
225:   getintvec($11,oldvec11);
226:   setintvec($11,@own_11);
227:   sec:=round(initvalue/18.2);
228:   min:=sec DIV 60;
229:   sec:=sec MOD 60;
230:   writeln; writeln('... successfully installed: ',
231:     'Display will switch off after ',min,':',sec);
232:   write('INT 10h - Service is ');
233:   IF int10 THEN writeln('activ!') ELSE writeln('not activ!');
234:   writeln;
235:   keep(0);
236: END.
237:
```

```
/-----------------------------------------------------------------------\

  1: title EXT.ASM
  2: ;
  3: ; Übersetzen mit MASM ext;
  4: ;
  5: status    equ    3bah           ; Display-Status-Port
  6: lp_set    equ    3b9h           ; Light-Pen-Flipflop setzen
  7: lp_reset equ     3bbh           ; Light-Pen-Flipflop löschen
  8:
  9: data      segment word public
 10:           assume ds:data
 11:           extrn switch:byte
 12:           extrn counter:word
 13:           extrn initvalue:word
 14:           extrn oldvec10:dword
 15:           extrn oldvec11:dword
 16:           extrn switch_boards:dword
 17: data      ends
 18:
 19: code       segment byte public
 20:            assume cs:code
 21:            public hercules
 22:            public lightpen
 23:            public own_10
 24:            public own_11
 25:            _ax dw 0
 26:            _bx dw 0
 27:            _cx dw 0
 28:            _dx dw 0
 29:            _bp dw 0
 30:            _si dw 0
 31:            _di dw 0
 32:            _es dw 0
 33:            _ds dw 0
 34:            _ss dw 0
 35:            _sp dw 0
 36:            vektor  dd 0
 37:            vektor1 dd 0
 38:
 39: hercules proc near
 40:           push  bp
 41:           mov   bp,sp
 42:           mov   dx,status       ; Display-Status-Port
 43:           mov   bl,5
 44:           in    al,dx
 45:           and   al,80h          ; Bit 7 isolieren
 46:           mov   ah,al
 47:           mov   cx,8000h
 48: test_loop:
 49:           in    al,dx           ; erneutes Lesen
 50:           and   al,80h          ; Bit 7 isolieren
 51:           cmp   al,ah           ; Änderung?
 52:           je    no_change
 53:           dec   bl              ; Bit 7 geändert
 54:           jz    ok_hrc          ; wenn sich Bit 7
 55:                                 ; fünfmal geändert hat,
 56:                                 ; wird die Schleife ver-
 57:                                 ; lassen
 58: no_change:
 59:           loop  test_loop       ; Erfolgt kein Wechsel
 60:                                 ; von Bit 7, liegt keine
 61:                                 ; Hercules-Karte vor
 62:           jmp   no_hrc
 63: ok_hrc:  mov   ax,01h          ; Ergebnis TRUE nach AX
 64:           jmp   stop
 65: no_hrc:  mov   ax,00h          ; Ergebnis FALSE nach AX
 66: stop:    pop   bp
 67:           ret
 68: hercules endp
 69:
 70: lightpen proc near
 71:           push  bp
 72:           mov   bp,sp
 73:           mov   dx,lp_set
 74:           out   dx,al
 75:           mov   dx,status
 76:           in    al,dx
 77:           test  al,2            ; Bit 1 (Statusport) = 0
 78:                                 ; Light-Pen-Flipflop nicht
 79:                                 ; gesetzt, Mode-Test nicht
 80:                                 ; möglich!
 81:           jz    end_lp
 82:
 83:           mov   dx,lp_reset
 84:           out   dx,al
 85:           mov   dx,status
 86:           in    al,dx
 87:           test  al,2
 88:           jnz   end_lp          ; Bit 1 (Statusport) = 1
 89:                                 ; Light-Pen-Flipflop gesetzt
 90:                                 ; -> Mode-Test nicht möglich
 91:           mov   ax,0001h        ; Ergebnis TRUE nach AX
 92:           jmp   end2
 93: end_lp:  mov   ax,0000h        ; Ergebnis FALSE nach AX
 94: end2:    pop   bp
 95:           ret
 96: lightpen endp
 97:
 98: own_10 proc far
 99:          mov cs:[_ax],ax
100:          mov cs:[_es],es
101:          mov cs:[_bx],bx
102:          mov cs:[_cx],cx
103:          mov cs:[_dx],dx
104:          mov cs:[_bp],bp
105:          mov cs:[_si],si
106:          mov cs:[_di],di
107:          mov cs:[_ds],ds
108:          mov cs:[_ss],ss
109:          mov cs:[_sp],sp
110:          mov ax,data                  ; Turbo Datensegment
111:          mov ds,ax                    ; nach DS
112:          mov ax,word ptr oldvec10     ; INT-10 Originalvektor
113:          mov word ptr cs:vektor,ax    ; ins Codesegment laden
114:          mov ax,word ptr oldvec10+2
115:          mov word ptr cs:vektor+2,ax
116:          cmp [switch],0               ; Bildschirm dunkel?
117:          jnz is_on                    ; nein! - dann nach is_on
118:          mov ax,0001h                 ; Parameter ON nach AX
119:          push ax                      ; und auf den Stack
120:          call dword ptr switch_boards; Bildschirm dunkel schalten
121:          mov [switch],01h             ; SWITCH auf ON setzen
122: is_on:  mov ax,[initvalue]           ; COUNTER neu initiali-
123:          mov [counter],ax             ; sieren
```

```
124:        mov bx,cs:[_bx]
125:        mov cx,cs:[_cx]
126:        mov dx,cs:[_dx]
127:        mov bp,cs:[_bp]
128:        mov si,cs:[_si]
129:        mov di,cs:[_di]
130:        mov es,cs:[_es]
131:        mov ds,cs:[_ds]
132:        mov ax,cs:[_ss]
133:        cli                             ; hier nicht stören!
134:        mov ss,ax
135:        mov sp,cs:[_sp]
136:        sti
137:        mov ax,cs:[_ax]
138:        jmp dword ptr cs:vektor         ; ab zum alten INT 10
139: own_10 endp
140:
141: own_11 proc far
142:        push ds
143:        push ax
144:        mov ax,data                     ; Turbo-Datensegment
145:        mov ds,ax                       ; nach DS laden
146:        mov ax,word ptr oldvec11        ; INT-11 Originalvektor
147:        mov word ptr cs:vektor1,ax      ; ins Codesegment laden
148:        mov ax,word ptr oldvec11+2
149:        mov word ptr cs:vektor1+2,ax
150:        pop ax
151:        pop ds
152:        pushf
153:        cli
154:        call dword ptr cs:vektor1       ; alten INT 11 aufrufen
155:        mov bx,0AAAAh                   ; Installationskennung
156:                                        ; nach BX
157:        iret
158: own_11 endp
159: code   ends
160:        end
```

***Bild 2** Externe Assembler-Unterprogramme für ADS.PAS*

wenn das Programm mit Hilfe von anderen residenten Utilities wie beispielsweise *Ramfrei* wieder aus dem Speicher entfernbar sein soll. Außerdem werden dadurch bereits installierte Interrupt-1C-Handler durch ADS nicht beeinflußt.

Der Tastatur-Interrupt-Handler

Im Tastatur-Interrupt-Handler wird über das Flag *SWITCH* geprüft, ob der Bildschirm dunkel geschaltet ist. In solch einem Fall wird er über den Aufruf der Prozedur *SWITCH_BOARDS* wieder aktiviert. Bei jedem Tastendruck wird prinzipiell der Zählerstand von *COUNTER* auf seinen Ausgangswert zurückgesetzt.
Tastatur- und Timer-Interrupt sind mit den Mitteln der Hochsprache programmiert (Bild 1). Obwohl dies auf den ersten Blick unproblematisch erscheint, sind Interrupt-Routinen in Turbo-Pascal genaugenommen nur eingeschränkt nutzbar. Der von Turbo-Pascal generierte Überbau für eine solche Interrupt-Routine benötigt einen übermäßig großen Stack und ein Umsetzen des Stackpointers auf einen eigenen Stack ist nicht vorgesehen. Wenn im Interrupt-Handler beispielsweise Stringfunktionen verwendet werden, so kann leicht ein Stackraum notwendig werden, der vom unterbrochenen Programm nicht in der Größe zur Verfügung gestellt wird.
COMMAND.COM reserviert beispielsweise nur etwa 150 Byte Stackraum. Hier kann es unter gewissen Bedingungen leicht zum Systemabsturz kommen. Prinzipiell sind die Konstruktionen der Hochsprache in ihrem Stackbedarf schwer zu kalkulieren (siehe dazu auch /1/). Da aber bei ADS die Tastatur- und Timer-Interrupt-Handler von simpelster Form sind, dürfte der Einsatz der Hochsprache hier gerechtfertigt sein.

Der Interrupt-10-Handler

Der Interrupt-10- ist wie der Interrupt-11-Handler als EXTERNAL-Prozedur deklariert (Bild 2). Hier soll die Symbiose aus Assembler und Hochsprache demonstriert werden. Der Interrupt-10-Handler funktioniert dabei in gleicher Weise wie der Tastatur-Interrupt-Handler. Eine interessante Aufgabe wäre es, diesen Interrupt-Handler mit den Mitteln der Hochsprache zu formulieren.

Der Interrupt-11-Handler

Der Interrupt-11-Handler dient dem Erkennen einer bereits existierenden Installation von ADS. Dieser Interrupt hat sonst die Funktion, die Hardwareausstattung des Rechners zu ermitteln und im AX-Register zu übergeben. Diese Funktion bleibt erhalten – nur liefert der Interrupt-11-Handler von ADS zusätzlich im Register BX den Wert OAAAAH als Zeichen für eine bereits erfolgte Installation zurück. Dadurch kann ADS sehr leicht prüfen, ob es bereits im Speicher vorhanden ist. Zwar sind für Anwenderprogramme die Interrupts 60H bis 66H vorgesehen, um jedoch die Integrität des residenten Programms zu sichern, wurde ein Interrupt gewählt, der mit einer zusätzlichen Funktion bestückt werden kann und dessen Aufruf zu beliebigen Zeiten keinen Einfluß auf das Systemverhalten hat. Dieses Kriterium wird beispielsweise auch vom Interrupt 12H erfüllt, der die Größe des Speichers ermittelt und im AX-Register zurückgibt.

Ermittlung der Grafik-Hardware

Der transiente Teil des Programms ermittelt unter anderem die Grafik-Hardware. Dazu wird zunächst auf der Speicherstelle 10H im BIOS-Datensegment (40H) das *Equipment-Word* nach dem Vorhandensein einer Monochrom-Karte abgetestet. Ist solch eine Hardware präsent, so kann es sich nur um einen Monochrom-Display-Adapter (MDA) ohne Grafikmöglichkeit oder um eine Hercules-Grafikkarte handeln. Ein expliziter Test auf das Vorhandensein der Hercules-Karte bringt die Gewißheit. Die dazu verwendete externe Assemblerroutine stellt eine Adaption aus /2/ dar. Maßgebliches Kriterium für die Existenz einer Hercules-Karte ist das Bit 7 des Display-Status-Ports (03BAH). Der MDA nutzt dieses Bit nicht, während sich sein Inhalt auf der Hercules-Karte dann zu Null ändert, wenn ein vertikaler Strahlrücklauf erfolgt. Bei einer Hercules-Karte wird noch zusätzlich das Vorhandensein eines Light-Pens mit Hilfe einer externen Assembler-Funktion nach /2/ getestet. Einerseits liefert dieser Test einen weiteren Anhaltspunkt für die Präsenz einer Hercules-Karte, andererseits ist es nur über den Ligth-Pen möglich, den Text- und Grafikmodus dieser Karte zu unterscheiden. Das Erkennen der Color-Karten gestaltet sich wesentlich einfacher. Hier muß nur zwischen CGA- und EGA-/VGA-Karte unterschieden werden. Wenn auf der Speicherstelle 0A8H im BIOS-Datensegment eine Null eingetragen ist, handelt es sich um eine CGA-Karte, im anderen Falle ist eine EGA- oder VGA-Karte wahrscheinlich. Da letztere Grafik-Hardware das BIOS ganz spezifisch mit neuen Funktionen erweitert, ist es möglich, über den Aufruf dieser erweiterten Funktionen die angeschlossene Hardware zu bestimmen. Ein Aufruf der Funktion 12H (AH=12H) mit BX=10H (EGA-Statusinformation anfordern) des Interrupts 10H gibt einen Anhaltspunkt für die Existenz einer EGA, wenn sich das BX-Register nach dem Aufruf verändert hat. EGA- und VGA-Karte lassen sich nach der gleichen Methode ab- und anschalten, so daß hier nicht weiter differenziert werden muß.

Dunkelschalten des Bildschirms

Das An- und Abschalten des MDA- und CGA-Bildschirms ist relativ leicht möglich. Beide Grafikkarten werden vom ROM-BIOS unterstützt. Das hat zur Folge, daß der Inhalt des Modusauswahlregisters (03D8H bei CGA, 03B8H bei MDA), der vom BIOS auf der Speicherstelle 65H im BIOS-Datensegment (CRT-Mode-Set) abgelegt wird, zur Manipulation direkt herangezogen werden kann. Durch Löschen des Bits 3 in besagter Status-Speicherstelle und Ausgabe des neuen Status am Port 03x8H wird der Bildschirm abgeschaltet. Ein einfaches Setzen dieses Bits und die Ausgabe über den entsprechenden Port sorgen dafür, daß der Bildschirm wieder zugeschaltet wird. Dieses Verfahren ist auch für die Hercules-Karte anwendbar, solange sie sich im Text-Modus befindet, denn nur dort verhält sie sich wie eine MDA-Karte. Der Grafik-Modus der Hercules-Karte wird vom BIOS nicht unterstützt, so daß das System nichts davon weiß, wenn diese Karte Grafik darstellt. Plausibel wird dieser Sachverhalt, wenn man weiß, daß die Umschaltung der Hercules-Karte vom Text- in den Grafik-Modus und zurück nur in eigener Regie und ohne Unterstützung des BIOS-Interrupts 10H erfolgen kann. Ebenso liefert die Funktion 15H (Get Mode) des BIOS-Interrupts 10H bei einer Hercules-Karte stets den Wert 7 zurück, unabhängig davon, ob sie im Grafik- oder Text-Modus ist.
Daher muß bei der Hercules-Karte zum An- und Abschalten ein anderer Weg gewählt werden, der direkt die Hardware anspricht und das BIOS damit umgeht. Die angewandte Methode beinhaltet das Abschalten des Bildschirms durch Nullsetzen der darzustellenden Zeilenzahl. Das Index-Register der Hercules-Karte 03B4H wird dazu mit dem Wert 6 geladen, wonach im Daten-Register 03B5H die Anzahl der anzuzeigenden Zeilen einzutragen ist. Das Anschalten des Bildschirmes geschieht analog – nur muß hierbei beachtet werden, ob die Hercules-Karte im Grafik- oder Text-Modus ist. Beide Modi unterscheiden sich durch unterschiedliche Zeilenzahlen (19H für Text, 57H für Grafik). Die Unterscheidung zwischen Text- und Grafikmodus erfolgt mit Hilfe des Light-Pen. Dazu wird der Light-Pen-Flipflop gesetzt und anschließend seine Adresse über den 6845-CRT-Controller abgefragt. Die Offset-Adressen im Text- und Grafik-Modus unterscheiden sich aufgrund unterschiedlicher Adreßschemen, so daß auf diese Art und Weise der

Modus der Hercules-Karte erkannt werden kann. Die EGA- und VGA- Karte wird nach einer in /3/ vorgeschlagenen Methode ab- und angeschaltet. Der funktionale Hintergrund dieses Verfahrens ist plausibel unter anderem in /4/ erklärt. Das Verfahren nutzt das Attribute-Adress-Register (AAR). Dieses Register dient sowohl als Index- als auch als Datenregister. Ein Lesen des Ports 3DAH (Farbmodi) beziehungsweise 3BAH (Monochrom) erklärt den Port 3COH definitiv zum Indexregister. Jede Ausgabeopertion auf diesen Port verändert zusätzlich ein internes Flipflop, das zwischen dem Index- und den Datenregistern umschaltet. Beim Schreibzugriff auf das Register 3COH wird mit den Bits 0..4 das konkrete Datenregister gewählt. Bei einer solchen Operation wird gleichzeitig der Bildschirm dunkel geschaltet. Verantwortlich dafür ist das Bit 5; ist es Null, so wird der Bildschirm abgeschaltet, im anderen Fall wird er wieder angeschaltet. Exakt dieser Sachverhalt wird bei der Lösung berücksichtigt.

Das Programm soll eine Anregung zur TSR-Programmierung sein. Mit ihm läßt sich das Spektrum an Möglichkeiten, das Turbo-Pascal für die TSR-Programmierung bietet, erkunden. Es trägt darüber hinaus durch die Beschäftigung mit der Registerprogrammierung der Grafikkarten zum tieferen Verständnis dieser leistungsfähigen Hardware bei. Wer engagiert ist, findet ein breites Betätigungsfeld.
Im Initialisierungsteil des Programms ist in Kommentarklammern der Aufruf des Interrupt 10H mit der Funktion 1AH enthalten. Er dient dazu festzustellen, ob eine echte VGA-Karte im System ist (AL=1AH). Dann könnte die Routine *Switch_8514* nach /5/ genutzt werden, die dort für einen 8514-Adapter Verwendung fand, aber auch einige VGA-Karten bedienen kann. Die Sache hat nur einen Haken: Einige EGA-Karten, die mit zusätzlichen Modi ausgestattet sind, verfügen unter anderem über die Funktion 1AH, so daß sie als VGA-Karten identifiziert werden. Da sie im allgemeinen jedoch nicht auch die Funktion AH=12H mit BL=36H unterstützen, kann es Systemabstürze geben.
Nach dem Schema der prozeduralen Parameter lassen sich Ab-/Anschaltroutinen für beliebige Grafikkarten in das Programm einfügen. So wäre eine Routine für die Grafikkarte des A 7150 eine sinnvolle Ergänzung. Ebenso wäre ein Verfahren interessant, mit dessen Hilfe diese Grafikkarte zweifelsfrei identifiziert werden könnte.

Literatur

/1/ Schäpers, A.: Turbo-Pascal 5.0. Addison-Wesley, Bonn 1989
/2/ Tilke, R.: Ist Hercules da? c't, Hannover (1988) 5, S. 128
/3/ Wilde, M.: EGA und VGA dunkelschalten. c't, Hannover (1989) 7, S. 121
/4/ Meiners, D.: Regie über Register. c't, Hannover (1987) 12, S. 231
/5/ Schmidt, T.: Interrupt-Vektor im Griff. mc, München (1989) 12, S. 58
/6/ Little, T.: Programmieren wie die Profis. DOS International, München (1989) 5, S. 170
/7/ Eickmann, I.: Der Bildschirm hat Pause. mc, München (1989) 9, S. 58
/8/ Little, T.: Speicherresidente Programme selbstgemacht. DOS International, München (1988) 2, S. 106

Standardgerechte Betriebssysteminterfaces

Norbert Frindt, Dr. Barbara Hackler, Dresden

Der Standardvorschlag IEEE 855

Die Möglichkeit des Einsatzes von Software auf Rechnern mit unterschiedlicher Architektur ist ein zentrales Problem der Informatik. Die Bereitstellung von Systemsoftware hält mit der Hardwareentwicklung noch schritt. Die Überführung von umfangreichen Anwendersystemen vollzieht sich dagegen in der Regel nur schwerfällig. Eine unproblematische Übernahme von Anwendungslösungen, die bereits auf anderen Computersystemen benutzt werden, ist meistens wünschenswert.
Systemunterschiede auf Maschinenbefehlsniveau können in vielen Fällen durch die Verwendung höherer Programmiersprachen überwunden werden. Die Verknüpfung der Anwendungssoftware mit dem Betriebssystem erschwert jedoch ihre Überführung auf eine Rechnerumgebung, die ein anderes Betriebssystem enthält. Eine Hauptursache für die Probleme bei der Überführung von Anwendersystemen ist in den Unterschieden bei der Parameterübergabe für Betriebssystemrufe zu suchen. Zur Überwindung dieser Schwierigkeiten wurde 1985 der Standardvorschlag IEEE 855 (MOSI, **M**icroprocessor **O**perating **S**ystems **I**nterface) des *Institute of Electrical and Electronics Engineers* (IEEE) veröffentlicht /1/. Er beinhaltet die Definition einer standardisierten Schnittstelle von Betriebssystemen. Dabei ist unerheblich, welches konkrete Betriebssystem die Schnittstelle liefert, vorausgesetzt, daß seine Funktionalität den Anforderungen entspricht.
Für bereits existierende Betriebssysteme kann beispielsweise durch die Entwicklung entsprechender Anpassungsmodule die standardisierte Schnittstelle zusätzlich bereitgestellt werden. Eine funktionelle Erweiterung des Betriebssystems wird dann notwendig, wenn es die Anwendungssoftware erfordert. Für die Zukunft bleibt zu hoffen, daß die Produzenten neuentwickelte Betriebssysteme von vornherein mit einem standardisierten Nutzerinterface ausstatten.
Die MOSI-Definition unterteilt die Gesamtheit aller Betriebssystemrufe in verschiedene Dienstkategorien (capabilities). Jede Dienstkategorie umfaßt ein oder mehrere Module. Ein solcher Modul enthält logisch zusammengehörige Funktionen. Die Kombination mehrerer Module wird als Konfiguration bezeichnet. Im Standard IEEE 855 wurden verschiedene Konfigurationen vorgeschlagen, eine nutzerspezifische Zusammenstellung ist jedoch möglich.
Ein Einsatzfall für MOSI kann die Schaffung von Software für lokale Netze sein – hierbei handelt es sich um spezielle Anwender- und Kommunikationssoftware. Es wird angestrebt, auf allen Rechnern eines lokalen Netzes einheitliche LAN-Software anzuwenden. Damit wird die Nutzung der Dienste des lokalen Netzes erleichtert. Ein wichtiges Kriterium für die Beurteilung von Modularität, Konfigurierbarkeit und Flexibilität einer LAN-Installation ist der Aufwand für die Einbindung heterogener Rechner oder Betriebssysteme.
Die Entwicklung einer Standardschnittstelle in der MOSI-Konfiguration *Echtzeitunterstützung* (real-time support) für 16-Bit-PCs und nachfolgend von standardorientierter LAN-Software war Gegenstand unserer Arbeiten zum Aufbau des heterogenen LANs NAGEMA-LOTUNET /2/. Im folgenden werden wesentliche Komponenten der Standarddefinition vorgestellt sowie am Beispiel der LAN-Software Bezüge zur Nutzung dieser Funktionen aufgezeigt.

Das MOSI-Multitaskkonzept

Die Grundlage für die Abarbeitung der LAN-Software LOTUNET ist die Nutzung eines Multitaskbetriebssystems /3/. Bei der Anwendung auf 8-Bit-Mikrorechnern mit dem nicht multitaskfähigen Betriebssystem CP/A werden im LAN NAGEMA-LOTUNET die entsprechenden Dienste des Echtzeitbetriebssystemkerns KOMI genutzt /4/.
Das Betriebssystem MS-DOS ist jedoch ebenfalls nicht multitaskfähig. Die mögliche Arbeit mit mehreren Programmen (Prozessen) ist prozedural organisiert. Es existieren Dienste zum Laden und zum Ausführen von Programmen. Die (nachgeladenen) Programme werden aber nicht (quasi-) parallel zum rufenden Programm, sondern als dessen Unterprogramme abgearbeitet. Nach Beendigung eines nachgeladenen Programms erhält der „Urheber" den Prozessor zurück – also der Prozeß, der den Ruf zum Laden ausführte. Danach kann er in seiner Abarbeitung fortfahren. Da eine solche Arbeitsweise dem LOTUNET-Konzept und der MOSI-Konfiguration *Echtzeitunterstützung* widerspricht, erwies sich die Nutzung einer Multitaskerweiterung als notwendig.

Das Prozeßmodell

Betriebssystemdienste zur Prozeßverwaltung können nicht losgelöst vom dazugehörigen Prozeßmodell betrachtet werden. MOSI definiert einen *Prozeß* als kleinste Einheit, die zur Ausführung (Abarbeitung) verwaltet werden kann. Dabei ist ein Prozeß die kleinste geschlossene Einheit, die den Prozessor beanspruchen kann. Es wird innerhalb der MOSI-Dienstkategorie *Prozeßverwaltung* zwischen der Arbeit mit einem und der mit mehreren Prozessen unterschieden.
Ein Prozeß kann Nachfolgerprozesse (Söhne) einrichten. Dazu dient der Dienst CREATE PROCESS. Prozesse haben im allgemeinen die Kontrolle über sich selbst und über ihre Söhne. Das bedeutet, daß sie die

angegebenen Prozesse zum Beispiel suspendieren, löschen oder resuspendieren können. Abweichungen von diesem Prinzip werden speziell ausgewiesen. MOSI betont, daß die Möglichkeiten der Einflußnahme eines Prozesses auf andere Prozesse sowie die möglichen Zustandsübergänge immer von den Scheduling-Algorithmen des jeweiligen Betriebssystems abhängig sind.
Beim Kreieren (Einrichten) eines Prozesses können ihm vom „Vater" Parameter übergeben werden. Solche Parameter sind zum Beispiel Identifikatioren für Semaphore, Files oder Mailboxen. Im Normalfall kreiert der Vaterprozeß alle von ihm und seinen Söhnen gemeinsam benutzten Datenstrukturen. Dabei erhält er als Funktionswerte die entsprechenden Identifikatoren.
Alle von einem Prozeß angeforderten Ressourcen (Files, Semaphore, ...) werden bei seiner Beendigung automatisch freigegeben. Dieser Umstand ist zu beachten, wenn solche Ressourcen auch von den Söhnen genutzt werden. In diesem Fall darf sich der Vaterprozeß erst beenden, wenn seine Ressourcen nicht mehr benötigt werden.
Im Standard IEEE 855 werden für einen Prozeß 5 mögliche Zustände definiert. Die erlaubten Zustandsübergänge der vorliegenden Programmversion sind in Bild 1 dargestellt. Die relativ große Anzahl erlaubter Übergänge ermöglicht einen variablen Einsatz der standardisierten Multitaskdienst-Schnittstelle für die unterschiedlichsten Anwendungen.

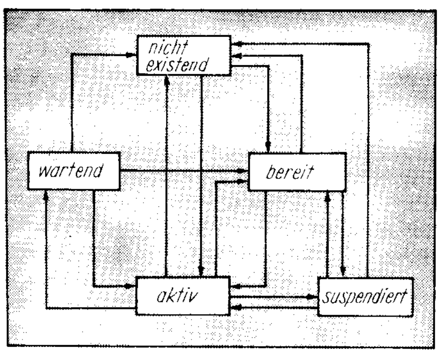


Bild 1 Prozeßzustände nach IEEE 855 und mögliche Übergänge bei der Nutzung des Betriebssystems SM-DOS

Dabei gelten folgende Definitionen der einzelnen Zustände:
– *nicht existent:* Der Prozeß ist dem System nicht (oder nicht mehr) bekannt.
– *bereit:* Der Prozeß wartet auf die Zuteilung des Prozessors.
– *aktiv:* Der Prozeß besitzt den Prozessor.
– *wartend:* Der Prozeß wartet auf ein Signal, eine Nachricht oder auf den Ablauf einer bestimmten Zeitspanne.
– *suspendiert:* Der Prozeß wartet auf seine Resuspendierung durch einen anderen Prozeß (nur Dienst RESUME PROCESS).

Mittel zur Prozeßsynchronisation

In Multitasksystemen ist es häufig notwendig, daß sich zwei oder mehrere Prozesse synchronisieren. Gründe dafür können beispielsweise die exklusive Nutzung eines Betriebsmittels (Drucker, File, ...), die Nutzung kritischer Abschnitte und anderes sein. Erfolgt die Synchronisation auf der Grundlage des Eintretens von Ereignissen (z. B. Belegen/Freigeben eines kritischen Abschnitts), so ist die Programmierung mit Hilfe von *Semaphoren* (Signalen) sehr effektiv möglich. Ein Semaphor besteht aus dem Wert einer Steuervariablen, die nur über spezielle Betriebssystemdienste geändert werden kann, und dem Anker einer Liste für die an dem entsprechenden Semaphor wartenden Prozesse. Der Semaphorwert wird um die Anzahl der eingetretenen Ereignisse korrigiert. Erst nach Eintritt der angegebenen Anzahl von Ereignissen wird ein Prozeß freigesetzt.
MOSI enthält Dienstaufrufe zur Nutzung einfacher und mehrfacher Semaphore. Ihr Unterschied besteht in der Registrierung des Eintretens eines oder mehrerer Ereignisse im Block. Für die Konfiguration *Echtzeitunterstützung* wird nur die Nutzung einfacher Semaphore gefordert, das heißt, es tritt immer nur ein Ereignis (Ankunft eines Signals) ein. Das Eintreffen mehrerer Signale wird in eine Folge von Einzelsignalen transformiert. Der auf die Signale wartende Prozeß hat dann eine Folge von WAIT SEMAPHORE-Rufen abzuarbeiten.
Für Semaphore existieren Dienste zum Kreieren (CREATE SEMAPHORE) und Löschen (DELETE SEMAPHORE). Das Senden von Signalen wird durch den Dienst SIGNAL SEMAPHORE, das Warten auf ein Signal durch WAIT SEMAPHORE ermöglicht.
Die häufig anzutreffende Unterteilung in binäre und nichtbinäre Semaphore wird bei MOSI nicht vorgenommen. Lauf Definition sind alle Semaphore nichtbinär. Das ist sinnvoll, da ein binäres Semaphor (mögliche Werte 0 und 1) auf ein nichtbinäres Semaphor unter Beachtung der Wertebereichsrestriktionen abgebildet werden kann.

Das Eintreffen von Signalen (SIGNAL SEMAPHORE) hat folgende Wirkungen:
1. Wartet ein Prozeß an dem Semaphor, wird er freigesetzt.
2. Wartet kein Prozeß, wird das Signal gespeichert, der Semaphorwert wird also inkrementiert.
Ein Prozeß hat die Möglichkeit, zeitbegrenzt oder unbedingt (starr) an einem Semaphor zu warten. Trifft in der angegebenen Zeit ein Signal ein, wird er freigesetzt. Anderenfalls erfolgt die Freisetzung nach Ablauf der Zeitspanne. Anhand des Funktionswertes von WAIT SEMAPHORE kann der Prozeß ermitteln, ob er ein Signal erhalten hat oder nicht. Das entspricht der Organisation von Timeout-Bedingungen.
Durch die Nutzung ereignisspezifischer Semaphore ist also eine reine Ereignissynchronisation einschließlich der Arbeit mit Timeouts möglich.
Bei der Abarbeitung des Rufes WAIT SEMAPHORE werden die Prozesse entsprechend ihrer Priorität in die Warteschlange des Semaphors eingeordnet. Dabei wird der Prozeß mit der höchsten Priorität als erster eingekettet, bei gleicher Priorität erfolgt die Einordnung nach dem FIFO-Prinzip. Die Warteschlangen werden bei Änderung der Priorität eines wartenden Prozesses (Dienst CHANGE PRIORITY) gegebenenfalls neu sortiert.
Erhält ein Prozeß ein Signal, wird er je nach seiner Priorität *bereit* oder *aktiv*. Das gleiche Verhalten zeigen Prozesse bei erfolglosem Warten nach Ablauf der angegebenen Wartezeit.

Mittel zur Prozeßkommunikation

Der Austausch von Informationen zwischen Prozessen erfolgt durch die Arbeit mit Nachrichten. Solche Informationen können zum Beispiel Ergebnisse anderer Programme oder Namen von Files sein.
Nachrichten werden in lokalen Mailboxen (repositories) abgelegt oder von dort gelesen. MOSI definiert Dienste zum Kreieren (CREATE REPOSITORY) und Löschen (DELETE REPOSITORY) von Mailboxen sowie für das Senden (SEND MESSAGE) und Empfangen (RECEIVE MESSAGE) von Nachrichten.
Die Dienste der Nachrichtenübermittlung sind analog den Semaphordiensten nutzbar. Diese Analogie betrifft vor allem die möglichen Zustandsübergänge und die Time-out-Organisation.
Bei jedem Senden von Nachrichten werden diese in systeminterne Puffer, beim Empfangen in nutzereigene Bereiche kopiert. Die Länge und die Anzahl der Puffer bleiben dem Nutzer unbekannt. Die Länge der übertragbaren Nachrichten ist beliebig.
Eine wichtige Anwendungsmöglichkeit der Dienste zur Prozeßkommunikation ist beispielsweise die Nutzung von Mailboxen als Implementation der Dienstzugriffspunkte (SAP – service access point) des OSI-Referenzmodells. Die Nachrichten enthalten dort die Protokoll- und die Dienstdateieinheiten (PDU – protocol data unit, SDU – service data unit). Jedem SAP wird eine Mailbox zugeordnet.

Stand der Entwicklung

Das Betriebssystem SM-DOS stellt im wesentlichen eine Multitaskerweiterung von MS-DOS 3.20 dar /5/. Hierfür wurden von uns die Dienstkategorien *Prozeßverwaltung* und *Prozeßsynchronisation und -kommunikation* entwickelt. Damit wurde die Grundlage für ein komfortables Multitasksystem zur Nutzung durch LAN-Software geschaffen. Es bestehen keine funktionellen Abweichungen vom Standard; es können lediglich nicht alle geforderten Fehlerursachencodes erzeugt werden, da SM-DOS nur eine begrenzte Anzahl von Rückkehrcodes liefert.
Außer den genannten Modulen stehen MOSI-gerechte Dienste zur dynamischen Speicherverwaltung, zum Verzögern von Prozessen und zur synchronen File-Ein-/Ausgabe zur Verfügung.
Der Diensteumfang und die Parameterstruktur des Betriebssystems SM-DOS erwiesen sich als sehr gute Voraussetzungen für eine MOSI-Schnittstelle. So mußten für einige MOSI-Dienste lediglich Konvertierungen der Eingabeparameter und Rückkehrcodes programmiert werden.
Die Einbindung der MOSI-Dienste in das Betriebssystem MS-DOS konnte problemlos als Erweiterung der SM-DOS-Einbindung erfolgen.

Einbindung der MOSI-Erweiterung in SM-DOS

Die Dienste von MS-DOS, SM-DOS und MOSI nutzen die gleiche (physische) Schnittstelle (Softwareinterrupt 21H, Dienstcode im Register AX, Parameter in Registern). Fast alle Dienste von MOSI nutzen die Funktionen von SM-DOS und MS-DOS. Daraus resultiert die in Bild 2 dargestellte Nutzungskonzeption der bei allen Systemrufen gleichen physischen Schnittstelle (INT 21H).

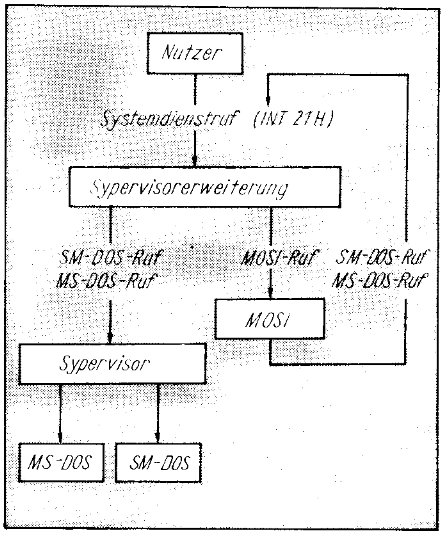


Bild 2 Nutzungskonzeption der Schnittstelle INT 21H

Durch die Supervisorerweiterung werden alle Betriebssystemrufe anhand des Funktionscodes in MOSI-Rufe einerseits und SM-DOS- bzw. MS-DOS-Rufe andererseits unterschieden. Handelt es sich um einen MOSI-Ruf, werden die im Standard definierten Parameter in die Parameter von MS-DOS oder SM-DOS konvertiert. Anschließend wird der entsprechende Betriebssystemruf ausgeführt. Nach dem Beenden der Ausführung erfolgt die Rückkonvertierung der Fehlercodes, der Resultatparameter und des Funktionswertes, soweit vorhanden.
Wie die Dienste von SM-DOS und MS-DOS erbracht werden, spielt für die Betrachtung aus Nutzersicht keine Rolle. In der vorliegenden Version stehen dem Nutzer neben den MOSI-Diensten alle Dienste der Systeme MS-DOS und SM-DOS zur Verfügung.

Literatur

/1/ IEEE Trial-Use-Standard specification for Microprocessor Operating Systems Interface (IEEE Std. 855). Institute of Electrical and Electronics Engineers, Inc., Dec. 1985

/2/ Löffler, H.; Grupe, R.: Das lokale PC-Netz NAGEMA-LOTUNET. Rechentechnik/Datenverarbeitung, Berlin 26 (1989) 2, S. 23

/3/ NAGEMA-LOTUNET Version 1.0, Anleitung für den Nutzer, Technische Universität Dresden, Informatik-Zentrum, 1988

/4/ Anwenderbeschreibung für das Echtzeit-Betriebssystem KOMI VM 3.3 für Mikrorechner mit den Mikroprozessoren U 880 D und Z 80. Technische Universität Dresden, Sektion Informationsverarbeitung, 1980

/5/ Liebeck, A.: SM-DOS-Programmierhandbuch. Technische Universität Dresden, Informatik-Zentrum, 1988

☒ KONTAKT ☎

Technische Universität Dresden, Informatik-Zentrum des Hochschulwesens, Bereich Rechnersysteme, Mommsenstraße 13, O-8027 Dresden; Tel. 4 57 53 80

Datenverwaltung mit dBase-kompatibler Software

Bernd Spitzhofer, Dresden

Die Arbeit mit Datenbankbetriebssystemen ist zu einem der Haupteinsatzfälle bei der Nutzung von Computertechnik geworden. Gerade im PC-Bereich haben sich Datenbankbetriebssysteme als Standardsoftware fest etabliert und eine erstaunliche Leistungsfähigkeit erreicht, die oft nicht voll ausgeschöpft wird. Es ist das Ziel des vorliegenden Beitrages, dem Nutzer durch Wertung einzelner Komponenten von dBase-kompatibler Software und durchgeführten Literaturvergleichen eine Hilfestellung zum effektiven Umgang hiermit zu geben.

Datenmodell

Wesentlich für die Arbeit mit einer Datenbank ist das ihr zugrunde liegende Datenmodell. Im Bereich der Mikrorechner hat sich besonders das relationale Datenmodell durchgesetzt, worauf die Untersuchungen beschränkt bleiben sollen, ohne damit anderen Modellen die Berechtigung abzusprechen. International werden momentan mehr als 60 solcher Softwareprodukte angeboten, die unter dem Betriebssystem MS-DOS laufen. Aus unterschiedlichen Gründen haben Datenbankbetriebssysteme der dBase-Familie besonders starke Verbreitung gefunden und dienen Herstellern als Vorlage, so daß ähnliche Kommandos verwendet, diese compiliert oder in ihrem Sprachumfang kompatibel gestaltet werden.
Vor dem Einsatz der entsprechenden Software ist es eine wesentliche Aufgabe des Nutzers, seine aufzubauende und auszuwertende Datenstruktur dem Datenmodell anzupassen: Es müssen komplizierte Datenstrukturen in einfache Tabellen aufgelöst werden, wobei gleichzeitig eine Normalisierung anzustreben ist. Entsprechend der Coddschen Theorie sind Redundanzen so zu beseitigen, daß mengenalgebraische Operationen auf den Datenbestand anwendbar werden, also logisch einwandfreie Datenstrukturen vorliegen (vgl. dazu /1/). Einfach prüfbare Kriterien sind die Vermeidung von identischen Datensätzen (Records) und die Bedeutungslosigkeit der Reihenfolge der Records. In der Praxis übersieht man oft diese wichtige Voraussetzung für ein effektives Arbeiten mit dem Datenbankbetriebssystem, nicht zuletzt deshalb, weil der Nutzer glaubt, seine bisher geführte Kartei einfach auf eine Datenbank übertragen zu können. Auf diesen fatalen Irrtum, den man bei der schematischen Übertragung solcher Strukturen begeht, weist /2/ hin. Das Einrichten von Relationen in Form von Lexika, also 1:1-Abbildungen, ist oft der erste Schritt, um Redundanzen zu beseitigen. Die Relationen ermöglichen es, im Record einen Code zu führen.
Vorteile der Normalisierung sind u. a.
- Platzeinsparung bei der Recordlänge
- einfacher Änderungsdienst
- mehrere Sichten auf die Datenbank
- erleichterter Datenschutz.

Softwarekomponenten

Wurde die nötige Klarheit über die Datenstruktur erzielt, gilt es, die Vorzüge und Nachteile der Softwareprodukte zu bewerten und ein entsprechendes System auszuwählen. Da eine komfortable Bedienung erst an 16-Bit-Computern möglich wird, soll auf die 8-Bit-Version dBase II nicht eingegangen werden; auch sind in /3/ wichtige Parameter im Vergleich zu dBase III klar dargestellt. Im folgenden werden deshalb dBase III und dBase III Plus kurz mit dBase bezeichnet.
Als Nutzerkriterien werden die *Benutzeroberfläche*, die *Abfragesprache*, die *Datenrepräsentation* und die *Schnelligkeit* bewertet.

Benutzeroberfläche

Abgesehen von den Erlebnissen bei der Installation eines Softwarepaketes, hat man durch die angebotene Benutzeroberfläche sein erstes Aha-Erlebnis, das wesentlich zur Akzeptanz des Systems beiträgt. dBase bietet durchgängig eine Pull-down-Menütechnik an, die durch eine Statuszeile ergänzt werden kann. Damit stehen dem Nutzer notwendige Informationen in einer sehr übersichtlichen Form als Hilfestellung bei der Arbeit mit dem System zur Verfügung. Besonders für eine interaktive Arbeitsweise ist diese Oberfläche vorteilhaft. Solche Befehle wie

CREATE QUERY	(Filterbedingungen definieren)
SET CATALOG	(Dateifilter benutzen)
CREATE VIEW	(Dateien verknüpfen)
CREATE SCREEN	(Bildschirmmaske definieren)

bestechen neben ihrer Leistungsfähigkeit durch ihre gute Menüführung, wobei die wesentlichsten sichtbaren Neuerungen gegenüber früheren dBase-Produkten aufgelistet sind. Hervorzuheben ist die Möglichkeit der Arbeit mit einem Maskengenerator. Hier entstehen durch einfache Menüauswahlen neben dem SCR-File ein FMT-File und ein TXT-File. Das FMT-File enthält eine Folge von @ ... SAY- und @ ... GET-Befehlen, die der Nutzer, ohne Vorhandensein dieser Benutzeroberfläche, umständlch zu einer solchen Maskendatei zusammenstellen müßte. Wer diese Arbeit bei früheren dBase-Versionen durchführen mußte, wird diese Möglichkeit, die durch die Befehle CREATE/MODIFY SCREEN aktiviert wird, dankbar nutzen, wobei gleichzeitig die Prüfung von Wertebereichen mit festgelegt werden kann.
Selbstverständlich bieten alle Softwarehersteller Hilfetexte an, die einen Überblick über den Befehlsvorrat und die Funktionen anbieten. Wichtig für die Handhabbarkeit ist, daß diese Texte ausgeblendet werden können, da sie dem versierten Nutzer hinderlich sind. Bewährt als Führung hat sich auch das ASSIST-Konzept, wie es dBase bietet.
Eine recht interessante, vom Nutzer selbst zu gestaltende Oberfläche bietet das Programmpaket FoxBase+ bei der Menüverwaltung an. Mit einfachen Befehlen, auf die ich noch im nächsten Punkt eingehen werde, ist ein Lichtbalkenmenü für die Programmsteuerung realisierbar. Bei Wertung der Benutzeroberfläche sollte im Vergleich mit Foxbase+ bemerkt werden, daß in der Version 2.0 keine ASSIST-Oberfläche und kein Maskengenerator innerhalb des FoxBase-Softwarepaketes angeboten werden. Diese effektiven

Hilfsmittel sind erst in der Version 2.1 mit Fox-Central und FoxView verfügbar, wobei letzteres wesentlich mehr als ein Maskengenerator für den Entwickler darstellt.

Abfragesprache

Durch die Verwendung einer Hochsprache zur Auswertung der Datenbestände wird eine wichtige Schnittstelle für den Nutzer geschaffen. Die Softwareprodukte unterscheiden sich in ihrem Sprachumfang, nehmen aber die dBase-Befehle als Sprachkern auf und erweitern diesen. Oft hat man den Eindruck, mit einem Original-dBase zu arbeiten. Die Erweiterungen des Sprachumfanges sind dem Nutzer oft nicht bewußt. Das System Quicksilver beispielsweise bietet die Möglichkeit des Einsatzes von benutzerdefinierten Funktionen und Windowtechnik; FoxBase erlaubt, 2dimensionale Variablen zu verwalten. Damit kommt das Problem der indizierten Variablen zum Wegfall. Zwei weitere, sehr effektive Befehlsgruppen sind bei diesem Produkt die Befehle der SYS-Gruppe, die Abfragen an das System ermöglichen und weitere Funktionen ausführen können (SYS(20) der deutschen Version auf Zeichenketten angewendet, garantiert eine korrekte Sortierung der deutschen Umlaute) sowie die Kombination von PROMPT- und MENU TO-Befehlen, die die Arbeit in einem Auswahlmenü mit Hilfe von Cursortasten gestatten. Die einzelnen Befehlsvorratserweiterungen sind sehr unterschiedlich und können hier nur angedeutet werden. Es steht aber außer Frage, daß fortgeschrittene dBase-Nutzer in Situationen kommen können, wo die Abfragesprache für das spezielle Problem nicht effizient ist, so daß eine andere Software eingesetzt werden muß. Dabei werden Programme mit dem gleichen Sprachkern bevorzugt.
Der Vielfalt der Befehle sollte durch die Schaffung eines Standards für die Sprache entgegengewirkt werden. Durch die einheitliche Abfragesprache SQL (Structured Query Language), die ebenso auf Großrechnern Berücksichtigung findet, wird diese Schnittstelle vereinheitlicht. dBase IV realisiert dieses Konzept.
Neben der Kenntnis von Spracherweiterungen ist es für den Nutzer vorteilhaft, sich die Wirkungsweise einzelner Befehle zu verdeutlichen, um so mehr, da es mehrere Strategien – und damit auch mehrere Befehle – gibt, um ein und dasselbe Rechercheergebnis zu erhalten. Grundsätzlich sollten für Suchprozesse keine sequentiell wirkenden Befehle benutzt werden. Bei dBase gilt es daher, mit Index-Files zu arbeiten und die damit zur Verfügung stehenden Befehle und deren Kombinationen einzusetzen. Das vorliegende logische Konzept legt ohnehin die Schaffung von Schlüsselfeldern nahe. Die Mächtigkeit solcher Befehle zeigt sich nicht zuletzt in der erzielten Schnelligkeit. Ein entsprechendes Beispiel wird deshalb im folgenden aufgeführt.

Datenrepräsentation

Die Akzeptanz eines Softwareproduktes wird auch durch die gebotenen Möglichkeiten der Darstellung von ausgewählten Daten der Datenbank bestimmt. dBase stellt einen Reportgenerator zur Verfügung, der die Listenerzeugung vereinfacht. Das kann sehr universell geschehen, da eine unmittelbare Verbindung zwischen Datenfile und Reportfile erst zum Laufbeginn hergestellt wird.
Die Verwendung von grafischen Ausgabeelementen ist dagegen sehr mangelhaft. Durch @-Befehle mit Koordinatenangaben läßt sich die Darstellung von Balken bei dBase lösen; FoxBase+ bietet zusätzlich den BOX-Befehl an, der auch Markierungsmuster zuläßt. Diese bescheidenen Mittel bedürfen einer Erweiterung. Dies ist durch die Nutzung von Grafik-Tools, die vom Hersteller angeboten werden und einer nachträglichen Installation bedürfen, erreichbar (vgl. /4/). Stehen diese Routinen nicht zur Verfügung, sei der Einsatz einer anderen Sprache empfohlen. Es ist innerhalb von dBase problemlos möglich, compilierte Programme, die zum Beispiel in Turbo-Pascal Version 4 geschrieben wurden, aufzurufen, wobei mit der Unit Graph ein komplettes Grafikpaket zur Verfügung steht. Der Zugriff auf die Datenbank erfolgt mit Pascal-Formulierungen, wobei vorher die dBase-Feldelemente in ein File mit Standardformat (Zusatz SDF) kopiert werden sollten.
Besteht der Wunsch, auf einzelne Variablen, die keine Feldelemente sind, zugreifen zu können, kann man ein Memory-File nutzen. Es ist generell so aufgebaut, daß beginnend mit dem 33. Zeichen die Variable abgelegt ist. Diese Variable ist für die Ausgabe in Überschriften oder für Achsenbezeichnungen oft wichtig, wobei dies gleichzeitig im Grafikmodus geschehen kann. Die Bilder 1 und 2 sind ein Beispiel für die grafische Datenausgabe von Werten einer dBase-Datenbank aus Säulen, wobei während der Sitzung die Überschrift, die dann als Variable übergeben wird, eingegeben werden kann.

Schnelligkeit

Ein Aspekt, der bei der anfänglichen Nutzung eines Softwareproduktes oft eine untergeordnete Rolle spielt – sei es nun aus Gründen des noch geringen Datenbestandes oder der noch nicht zur Routine gewordenen Anfrage an das System –, ist die Geschwindigkeit der Abarbeitung. Es kommt aber jeder Nutzer an den Punkt, wo sein Programm zu langsam ist. Es sei hier gleich bemerkt, daß der „reine dBase-Nutzer" sich zuerst mit dieser Fragestellung auseinandersetzen muß. Eine größere Geschwindigkeit läßt sich für dBase-Programme auf 3 Wegen erreichen:

Einsatz einer anderen Hardware
Die Einflüsse, die unter anderem das Vorhandensein einer Festplatte oder die Taktfrequenz des Prozessors auf die Geschwindigkeit ausüben, sind in /5/ sehr gut aufgelistet. Wir wollen an dieser Stelle davon ausgehen, daß der Nutzer sich auf einen Rechnertyp mit einer bestimmten Konfiguration festlegen mußte und bewerten die weiteren 2 Möglichkeiten intensiver.

Verwendung von Compilern
Ein wesentlicher Nachteil des dBase-Konzeptes ist die Nichtexistenz eines Compilers innerhalb dieses Programmpaketes. Führt man eine Übersetzung des Quellcodes durch, ergeben sich wesentlich kürzere Abarbeitungszeiten. Häufig wird der Clipper-Compiler verwendet. Dabei treten aber unter Umständen beträchtliche Compilierungszeiten auf, und die resultierenden EXE-Files haben stattliche Größen. Bezieht man sich auf /6/, wo unterschiedliche Tests im Zeitverhalten beurteilt werden, so schneidet FoxBase+ im Vergleich zu dBase – etwa 6,5 mal schneller – am besten ab. Meine eigenen Beobachtungen bestätigen dies durch ähnlich gute

```
program dbgraf; {Programm für dBASE-Grafik
                 Beispiel der Variablenübergabe
                 und Ausgabe von Balken}
uses graph,crt;
procedure trans(datana: string; var str: string);
var
 ch: char;
  r: integer;
aus: file of char;
 st: array[0..50] of char;
     begin
     Assign(aus,datana);reset(aus);
     str:='';
     r:=0;
     while not eof(aus) do
           begin;
           r:=r+1;
           read(aus,ch);
           st[r]:=ch;
           end;
     close(aus);
     {bei Textübergabe von max. 12 Zeichen}
     for r:=33 to 33+11 do str:=str+st[r];
     end;
const
x0 =  30;
y0 = 160;
var
graphmode,graphdriver,r:integer;
filna,sing : string;
ch: char;
 f: text;
{bei 10 übergebenen Feldelementen}
y: array[0..9] of integer;
    begin
    graphdriver:=detect;
    initgraph(graphdriver,graphmode,'');
    filna:='text1.mem';
    trans(filna,sing); {Textübernahme aus dBASE}
    OutTextXY(60,2,' Wertedarstellung als '+sing);
    Assign(f,'balkwert.dat');reset(f);
    r:=0;
      while not eof(f) do begin r:=r+1;
      read(f,y[r-1]); {Übernahme der Feldelemente}
      end;
    close(f);
    for r:=1 to 10 do
    bar3d(x0+(r*20),y0-y[r-1],x0+20+(r*20),y0,10,TopOn);
    ch:=readkey;
    closegraph;
  end.
```

***Bild 1** Beispiel einer Ausgabe im Grafikmodus bei Übernahme des Variableninhaltes aus einem dBase-File*

```
* PROGRAMM 3D-BALKEN
* es zeichnet Balken entsprechend der
* Werte des Feldes wert
use balken
copy to balkwert.dat fields wert sdf
store "              " to text
@ 5,10 say "Eingabe der Darstellungsform"
@ 6,10 say "(Balken, Kreis usw.)" get text;
 picture "AAAAAAAAAAAA"
* laden einer Variablen für überschrift
read
save to text1 all like text
run dbgraf
* eof
```

***Bild 2** Beispiel des Aufrufes eines compilierten Pascal-Programms innerhalb von dBase*

Werte, die nachfolgend noch aufgelistet werden. Besonders günstig bei der Nutzung von FoxBase ist die Tatsache, daß ein integrierter Compiler existiert. Die Compilierungszeit des dBase-Quellcodes ist unwesentlich, beispielsweise für ein 70 KByte großes File unter einer Minute. Bei der Arbeit mit dem Precompiler wird die Existenz des FOX-Files erkannt, was die Abarbeitungszeit verkürzt. FoxBase+ muß in dieser Hinsicht als eine gelungene dBase-kompatible Software angesehen werden (siehe auch den Beitrag auf Seite 24).
Steht kein Compiler zur Verfügung, ist die dritte Möglichkeit zu prüfen.

Programmiertechnik
Es sei dem Nutzer empfohlen, besonders beim Anwachsen seiner Daten, über die zeitlichen Abläufe bei der Programmabarbeitung nachzudenken und ihre Unabänderlichkeit in Frage zu stellen. Auf die Verwendung von Index-Files wurde schon hingewiesen. Die Bedeutung dieser Dateien liegt in einer schnelleren Reaktionszeit und im Vorliegen einer sortierten Folge. Unter Beachtung dieser beiden Aspekte ist ein effektives Programmieren möglich. Dies sei am Beispiel einer Auszählaufgabe erläutert: Will man in einem Feld die Häufigkeit einer bestimmten

```
* PRG-File mit Namen SLOW
* Auszählen eines Codes - sequentiell
use test
count for code=1 to a1
 ...
 ...
count for code=9 to a9
 ...
 ...
* eof

* PRG-File mit Namen QUICK
* Auszählen eine Codes - Verwendung einer Indexdatei
use test
index on code to codetemp
use test index codetemp
goto top
find 1
count while code=1 to a1
 ...
 ...
count while code=9 to a9
 ...
 ...
* eof
```

Bild 3. Programmvarianten zur Auszählung eines Codes

Kodierung auszählen, ist es ab einer bestimmten Recordanzahl und -größe günstiger, zunächst eine Indexdatei zu bilden und dann in dieser sortierten Datei den ersten Datensatz zu finden, auf den die Bedingung zutrifft. Dieser Zählvorgang muß mit der WHILE-Bedingung optimiert werden. Bild 3 zeigt die beiden Befehlsfolgen, die 10 Kodierungen in einem Feld auszählen. In Tafel 1 sind die Ergebnisse dargestellt, die als Mittelwerte anzusehen sind und unter Nutzung eines A 7150 mit Festplatte unter dem Betriebssystem DCP ermittelt wurden. Aus der Tafel ist nicht nur das Zeitverhalten bei verschiedenen Programmiertechniken, sondern auch der Zeitgewinn bei Einsatz eines Compilers zu entnehmen. Ab einer Dateigröße von 100 Byte Recordlänge und einer Recordanzahl von > 200 lohnt es sich, nach alternativen Lösungen zu suchen. Bei größeren Dateien von 300 Byte und 1000 Records liefert

Tafel 1 Zeitvergleich bei Zählzyklus

Record		dBase III Plus			FoxBase+				
Länge (Byte)	Anzahl	slow	quick	Fakt zu 3	slow	Fakt zu 3	quick	Fakt zu 4	Fakt zu 3
1	2	3	4		5		6		
100	250	0:37	0:25	1,5	0:11	3,4	0:08	3,1	4,6
100	500	1:13	0:46	1,6	0:23	3,2	0:15	3,0	4,9
100	1000	2:20	1:26	1,6	0:44	3,2	0:27	3,2	5,2
200	250	0:51	0:38	1,3	0:12	4,2	0:12	3,2	4,2
200	500	1:41	0:51	2,0	0:25	4,0	0:21	2,4	4,8
200	1000	3:24	1:55	1,8	1:18	2,6	0:36	3,2	5,6
300	250	1:08	0:45	1,5	0:14	4,9	0:14	3,2	4,9
300	500	2:16	1:16	1,8	0:50	2,7	0:21	3,6	6,5
300	1000	4:30	2:13	2,0	1:35	2,8	0:42	3,2	6,4

Spalten 4 und 6 sind incl. der Index-Datei-Bildungszeit
Spalten 5 und 6 sind incl. der Compilierungszeit

die Programmiermethode, die eine sequentielle Suche umgeht, eine Schnelligkeitssteigerung auf das Doppelte. Der erzielte Zeitgewinn mit FoxBase+ bei diesem Test, wo so vorgegangen wurde, als wäre dBase geladen, muß natürlich geringer als zitiert ausfallen. Er ist aber selbst unter diesen nicht optimalen Bedingungen (Hauptspeicher 512 KByte, kein bereinigtes Directory, Compilierungszeit mitgerechnet) eine 3fache Steigerung der Geschwindigkeit feststellbar.
Für Nutzer von dBase III sind diese Ergebnisse nicht anwendbar, da hier die Bildung von Indexdateien noch sehr zeitintensiv ist. Es kann nachgewiesen werden, daß durch geeignete Softwarewahl und deren Ausschöpfung eine wesentliche Leistungssteigerung bei der Nutzung dBase-kompatibler Systeme erzielt wird. Je nach Aufgabenstellung sollte man die Stärken (Bedienerführung, Schnelligkeit) einzelner Komponenten nutzen, die sich ineinander überführen lassen.

Literatur

/1/ Grill, E.: Relationale Datenbanken. München: AIT-Verlag, 1987
/2/ Pape, U.: Datenbanken anlegen – aber wie? Mikroprozessortechnik, Berlin 2 (1988) 8, S. 232
/3/ Kruczynski, K.: Von dBase II nach dBase III. Rechentechnik/Datenverarbeitung, Berlin 25 (1988) 6, S. 24
/4/ Cooper, J. T.: Tools für dBase III Plus: Grafik. Haar: Markt & Technik Verlag, 1987
/5/ Grafik, W.; Osten, B.: dBase III im Vergleich. Mikroprozessortechnik, Berlin 2 (1988) 10, S. 294
/6/ Mierzowsky, T.: Familienzuwachs dBase-III-kompatible Datenbanksoftware. c't, Hannover 4 (1987) 7, S. 68

⊠ KONTAKT ☏

Bezirks-Hygieneinspektion und -institut Dresden, Reichenbachstraße 71–73, Dresden, 8020; Tel. 47 87 52

Das Aids-Programm

Bernd Schewc, Gardelegen

Dieser Bericht veranschaulicht Gefahren, die von Computerviren ausgehen können. Er basiert auf den Erkenntnissen einer Kurzprüfung der sogenannten AIDS-Diskette – auch „Panamavirus" genannt –, von der erst vor wenigen Wochen erste Kopien auftauchten. Die hier gemachten Aussagen erheben keinen Anspruch auf Vollständigkeit. Um es gleich vorwegzunehmen: AIDS ist kein Computervirus im herkömmlichen Sinne.

Der Inhalt

Auf der Diskette befinden sich zwei Programme, die zunächst einen recht harmlosen Eindruck hinterlassen:

INSTALL EXE 146188 28.09.89 16:28
AIDS EXE 172562 07.08.89 10:40

Lediglich das Installationsprogramm mit 146 KByte fällt dem versierten Anwender ins Auge, und genau dieses Programm ist die Wurzel allen folgenden Übels. Der erste Versuch eines unbedarften Anwenders ist in den meisten Fällen wohl der direkte Aufruf der AIDS.EXE, der jedoch mit der Meldung: „You must run the INSTALL programm before you can use the AIDS programm" abgebrochen wird. Auch das Kopieren der Datei AIDS.EXE führt nicht zu dem gewünschten Erfolg. Ohne INSTALL läuft ganz offentsichtlich nichts, und INSTALL verlangt sofort die Entfernung des angebrachten Schreibschutzaufklebers, was für ein Sharewareprogramm doch ein wenig ungewöhnlich ist. Noch ungewöhnlicher ist die Tatsache, daß sich das einmal gestartete Installationsprogramm nicht mehr abbrechen läßt. Weder CTRL C noch ALT, CTRL und DEL ermöglichen es, das Programm zu verlassen. Hier hilft nur noch der Netzschalter oder die vom Programm geforderte Schreibschutzentfernung. Entschließt man sich hierzu, so beginnt INSTALL damit, AIDS auf die Festplatte zu kopieren. Nach Beendigung des Kopiervorgangs wird der Anwender freundlich, aber bestimmt aufgefordert, einen Betrag von $ 189,– für eine 1-Jahreslizenz (365 Anwendungen) oder von $ 378,– für eine 60-Jahreslizenz zu überweisen. Ein Zahlungsformular wird über den Drucker gleich ausgedruckt, und der Anwender aufgefordert, die Installationsdiskette an seine Freunde weiterzugeben. Nebenbei geschehen noch zahlreiche absonderliche Dinge, auf die jedoch erst später eingegangen werden soll. Jedenfalls ist es jetzt geschafft, mit dem Programm AIDS kann man sich nun über die Krankheit AIDS informieren. Abhängig von Wohnort, Alter, Sexualverhalten und allgemeinem Gesundheitszustand werden eine persönliche Risikoanalyse (Einstufung in eine der Gefährdungsklassen 1–7) erstellt sowie Verhaltensratschläge unterbreitet. Die böse Überraschung kommt nach zirka 3 Monaten, vorausgesetzt, der Rechner wird täglich genutzt und einmal täglich gebootet. Dann nämlich erscheint die Meldung: „Please wait 30 minutes during this operation. Warning-do NOT turn off the computer because you will damage the files on the hard disk drive. You will receive more information later." Jeder Versuch, den Vorgang zu unterbrechen (ALT,CTRL+DEL) führt zu dem Hinweis, daß alle Daten verloren gehen werden, wenn das Programm abgebrochen wird – mit eindringlichem Sirenengeheul. Nach einiger Zeit wird die Meldung geändert: „Insert the software RENEWAL DISKETTE or the USED SHARE DISKETTE into drive A and then press ENTER or RETURN."
Sollte man wider Erwarten (man wurde ja vor 3 Monaten zur Weitergabe aufgefordert) dennoch über die Installationsdiskette verfügen und diese in Laufwerk A einlegen, so wird diese gelöscht, und der Rechner druckt das bereits bekannte Zahlungsformular aus. Wird nun von der Festplatte neu gebootet, so schein zunächst alles normal. Aber vor jedem C: Prompt weist das System darauf hin, daß es gefährlich ist, den Rechner weiter zu nutzen, da die Datenbestände gefährdet sind. Weiterhin akzeptiert das Betriebssystem nur

```
Verzeichnis C:\
      C:\IO.SYS
      C:\MSDOS.SYS
      C:\COMMAND.COM
Verzeichnis C:\DOS
      C:\DOS\_RCC}1@. AK
      C:\DOS\}1&#. BZ
      C:\DOS\}XX81@. AK
      C:\DOS\}&&#´1. AK
      C:\DOS\}66H#U. AU
usw...
Verzeichnis C:\
Verzeichnis C:\ \
Verzeichnis C:\ \ \
Verzeichnis C:\ \ \ \ERROR IN.THE
      C:\ \ \ \_____. __
      C:\ \ \ \__. _
      C:\ \ \ \____.  _
      C:\ \ \ \_.  _
      C:\ \ \ \_____.  _
      C:\ \ \ \___.  _
      C:\ \ \ \___. _
      C:\ \ \ \____. _
      C:\ \ \ \___. __
      C:\ \ \ \_____. _
Verzeichnis C:\
      C:\\REM.EXE
      C:\\CYBORG.EXE
      C:\\AUTOEXEC.BAT
      C:\\AUTO.BAT

8486 nichtzusammenhängende Blöcke in 1 Bereichen gefunden!
Sollen nichtzusammenhängende Blöcke in Datein umgewandelt werden?
(J/N)? n
 17379328 Bytes Speicherplatz
können verfügbar gemacht werden!

 21239808 Bytes Gesamt-Disketten-/Plattenspeicher
  3846144 Bytes in 74 unsichtbaren Datei(en)
    12288 Bytes in 6 Verzeichnis(sen)
     2048 Bytes in 1 Benutzerdatei(en)
        0 Bytes verfügbar auf Diskette/Platte
   655360 Bytes Arbeitsspeicher
   609968 Bytes frei
```

***Bild 1** Inhaltsverzeichnis der Festplatte nach der Verschlüsselung*

noch Kommandos wie DIR und CLS, erlaubt jedoch keinen einzigen Programmaufruf, obwohl die Programme in der jeweiligen Directory stehen. Nach etwa zwei Minuten verabschiedet sich das System, und obige Meldung ist wieder zu lesen. Bei derlei Programmieraufwand ist es wohl unnötig zu erwähnen, daß der Bootvorgang ebenfalls nicht echt ist, sondern nur simuliert wird.
Will man also von der Diskette booten, so muß die Netzspannung abgeschaltet werden. Ein Blick auf das Inhaltsverzeichnis der Festplatte sorgt nun für einen gewaltigen Schock. Außer CYBORG.DOC mit einer Länge rund 1 KByte ist das gesamte Verzeichnis leer! Die Festplatte hingegen ist voll, es ist kein einziges Byte mehr frei. Ein Blick mit Hilfe der üblichen Utilities (Norton, PC-Tools etc.) offenbart Sonderbares. Die Directorystruktur ist noch voll erhalten, aber es sind neue Directories, deren Namen ausschließlich aus Leerzeichen bestehen, hinzugekommen. In diesen Directories befindet sich eine Datei, die für die Norton Utilities das Aus bedeutet. Die Datei trägt den Namen ERROR IN.THE. Wem es jedoch gelingt (z. B. CHKDSK /v) einen Blick in die neue Festplattenstruktur zu erhaschen, der findet die im Bild 1 dargestellte Struktur. Somit steht fest, was eigentlich geschehen ist. Sämtliche Dateinamen wurden nach einem bestimmten Algorithmus verschlüsselt und die Dateien, ebenso wie die Subdirectories, mit Hidden- und ReadOnly-Attribut versehen. Sodann wurden alle noch freien Cluster in nicht zusammenhängende Blöcke verkettet, so daß kein einziges Byte mehr frei bleibt.

Wie werd ich's wieder los?

Im Grunde gestaltet sich diese Arbeit verhältnismäßig einfach, wenn man davon ausgeht, daß jeder über diverse Hilfsmittel verfügt, mit denen Files umbenannt werden können. Denn die Verschlüsselung der Daten bezieht sich lediglich auf die Dateinamen, nicht auf den Inhalt. Nur, wer ist in der Lage, aus dem Dateiinhalt den Namen abzuleiten? Das wäre sicher ein wenig viel verlangt. Aber wenn der Rechner von der Festplatte gestartet wird, erscheint doch das Inhaltsverzeichnis mit allen Dateien. Also müssen die alten Directoryinformationen auch noch irgendwo auf der Platte stehen. Dem ist auch so, und man findet die alten Inhaltsverzeichnisse als ASCII-File in der Datei „— . –" im dritten unsichtbaren Subdirectory. Von dieser Datei kann ein Ausdruck gemacht werden, mit dessen Hilfe die Dateien wieder umbenannt werden können. Das ist zwar etwas mühsam, aber da meist nur einige wenige Dateien wirklich so wichtig sind, daß sie unbedingt restauriert werden müssen, ist dies ein gangbarer Weg, um einem großen Datenverlust vorzubeugen. Zusammenfassend läßt sich sagen, daß dieses Programm zwar mit „unfeinen" Mitteln arbeitet, jedoch diese Mittel nur zum Tragen kommen, wenn der Anwender mit dem Programm arbeitet, ohne Lizenzgebühren zu zahlen. Da hier jedoch überhaupt keine Möglichkeit zur Zahlung bestand – denn die ehrlichen Anwender, die versuchten, die Gebühren auf das angegebene Konto in Panama zu zahlen, erhielten ihr Geld postwendend zurück (Quelle: dpa) – wird man zwangsläufig mit der Verschlüsselung seiner Festplatte konfrontiert. Hinzu kommt, daß der Zähler von den Bootvorgängen und nicht vom Aufruf (also der Nutzung) des AIDS-Programms abhängig ist.

✉ KONTAKT ☏

BCS, Hard- und Softwareentwicklung, Bernd Schewc, Marktstraße 2, PSF 95, Gardelegen, 3570

AT-Laufwerk liest 720-KByte-Disketten

Christian Hanisch, Berlin

Das Lesen und Schreiben von 720-KByte-Disketten in 5.25-Zoll-Laufwerken des AT wird in der Regel über zusätzliche Driver ermöglicht. Daß auch eine recht pragmatische Lösung, die keinen zusätzlichen Driver benötigt, zum gewünschten Ziel führen kann, soll dieser Beitrag zeigen.

Gäbe es das 720-KByte-Format für 5.25-Zoll-Laufwerke am AT standardmäßig, so wäre kaum Raum gewesen für eigenprogrammierte Lösungen in diesem Punkt in zahlreichen Publikationen, die stets auch ein exklusiveres Verständnis für die interne Logik des DOS-Systems brachten. In diesem Licht ist denn auch die folgende Lösung zu sehen, die in Form eines residenten Programms das Lesen und Schreiben von 5.25-Zoll-Disketten im 720-KByte-Format im Laufwerk A: oder B: des AT gestattet. Auf interne Details soll hier weitgehend verzichtet werden. Diese werden umfassend unter anderem in /1/ und /2/ dargestellt. Es soll vielmehr nur ein Programm vorgestellt werden, dessen Einsatz insbesondere im Zusammenhang mit Backup/Restore im 720-KByte-Format zu sehen ist. Diese sogenannten externen DOS-Kommandos können nämlich logische 720-KByte-Laufwerke weder als Quelle noch als Ziel verwenden, wenn sie durch Driver bedient werden. Die Lösung scheint auf den ersten Blick gewagt, denn sie vertraut darauf, daß der AT bei jedem Diskettenzugriff über die Funktion *Read Change-Of-Disk Status* (16H) des Interrupt 13H zunächst prüft, ob eine Diskette gewechselt wurde. Eine Observierung des Diskettenzugriffs bestätigte, daß jedem Schreib-/Lesezugriff der Aufruf der Funktion 16H vorangeht. Ferner wird im Ergebnis dieser Prüfung, wenn es erforderlich ist, der Diskettenstatus auf der Speicheradresse 40H:90H für Laufwerk A: bzw. 40H:91H für Laufwerk B: – sofern physisch vorhanden – korrigiert. Beim Erkennen der einfachen Schreibdichte stellt das AT-BIOS stets auch den *Double-Stepping-Modus* (Bit 5 der Disk-Status-Zelle setzen) ein, der die Behandlung des 360-KByte-Formates im AT-Laufwerk ermöglicht. Da die Parameter für das 360- und 720-KByte-Format bis auf die Anzahl der Spuren identisch sind, kann durch einen Patch der Disketten-Status-Zelle das Double-Stepping abgeschaltet werden. Genau dies wird vom Programm AT720 bewältigt. Dazu stellt es einen Interrupt-13-Handler zur Verfügung, der die Funktion 16H und den Fehlerstatus einer Diskettenoperation überwacht. Zunächst wird vom Interrupt-Handler der alte Interrupt 13H aufgerufen. Wenn der Aufruf mit der Funktion 16H geschah, wird im Anschluß an die Ausführung des Originalinterrupts die Diskettenstatus-Zelle mit der Maske 0DFH *UND*-verknüpft. Außerdem prüft der Interrupt-Handler den Fehlerstatus des Diskzugriffs. Im Falle eines Fehlers wird das Doppel-Stepping wieder zugeschaltet (Bit 5 setzen) – in der Annahme, daß eine normale 360-KByte-Diskette im Laufwerksschacht liegt. So ist es möglich, 720- und 360-KByte-Disketten wechselseitig zu lesen und zu beschreiben. Das 1.2-MByte-Format steht dabei nach wie vor zur Verfügung.

Bei der Installation des Programms wird auf das Vorhandensein eines 5.25-Zoll-Laufwerkes mit 1.2 MByte geprüft. Dazu wird zum einen über die Funktion 15H des Interrupt 13H festgestellt, ob das Laufwerk ein *Disk-Change-Signal* unterstützt, und zum anderen werden die Informationen aus dem CMOS-RAM genutzt, um die Art des Lauf-

werkes zu ermitteln. Die Laufwerkstypen stehen im Register 10H des CMOS-RAM des AT. Das obere Halbbyte charakterisiert Laufwerk A, und die unteren vier Bit sind für Laufwerk B zuständig. Es stehen insgesamt vier Laufwerkstypen zur Auswahl:

Typ 1: 0001 – 360 KByte, 5.25 Zoll
Typ 2: 0010 – 1.2 MByte, 5.25 Zoll
Typ 3: 0011 – 720 KByte, 3.5 Zoll
Typ 4: 0100 – 1.44 MByte, 3.5 Zoll

Von besonderem Interesse bei der programmtechnischen Lösung dürften noch die Inline-Makros sein, die das Einsetzen der Originalregisterwerte vom Interrupt-Stapel und das Modifizieren der Pseudoregister auf dem Interrupt-Stapel mit den Ergebniswerten vom Interrupt-Aufruf gestatten. Sie können so auch für andere Interrupts Verwendung finden.
Eine Bemerkung zum Schluß sei noch gestattet: Natürlich ist das Programm keine umfassende Lösung, was das Problem des 720-KByte-Formates anbelangt. So muß das Formatieren der 720-KByte-Disk auf andere Weise erfolgen (beispielsweise mit der in /4/ vorgestellten Lösung). Jedoch wird mit diesem Beispiel demonstriert, wie mit wenig Aufwand einem leidigen Problem am AT begegnet werden kann.

```
  1: PROGRAM AT720; (* Turbo Pascal 5.x *)
  2: {$M 1024,0,0}
  3: {$N-,S-,F- }
  4: USES dos;
  5: CONST ver = '2.5';
  6:       (* Alle Funktionen des INT 13h die in AH einen
  7:          Fehlerstatus liefern *)
  8:       fkt = [$01,$02,$03,$04,$05,$0a,$0b,$0c,$0e,$0f,
  9:              $10,$11,$12,$13,$14];
 10: VAR   regs                          : registers;
 11:       alt_13, vektor                : pointer;
 12:       diskstatus                    : ^byte;
 13:       drive                         : CHAR;
 14:       pstr                          : STRING[2];
 15:       disk,disks,disp               : BYTE;
 16:       drivea,driveb,
 17:       dcs,err,d,cancel              : BOOLEAN;
 18:
 19: PROCEDURE get_regs;
 20: { Prozessorregister mit den Werten des INTERRUPT-
 21:   Stapels laden                                   }
 22:   INLINE($8B/$46/$16              { mov   ax,[bp+16]  }
 23:          /$50                     { push  ax          }
 24:          /$9D                     { popf              }
 25:          /$8B/$46/$10             { mov   ax,[bp+10]  }
 26:          /$8B/$5E/$0E             { mov   bx,[bp+0e]  }
 27:          /$8B/$4E/$0C             { mov   cx,[bp+0c]  }
 28:          /$8B/$56/$0A             { mov   dx,[bp+0a]  }
 29:          /$8B/$76/$08             { mov   si,[bp+08]  }
 30:          /$8B/$7E/$06             { mov   di,[bp+06]  }
 31:          /$8E/$46/$02);           { mov   es,[bp+02]  }
 32:
 33: PROCEDURE set_regs;
 34: { INTERRUPT-Stapel mit den aktuellen Inhalten der
 35:   Prozessorregister versehen                      }
 36:   INLINE ($89/$46/$10             { mov   [bp+10],ax  }
 37:            /$9c                   { pushf             }
 38:            /$58                   { pop   ax          }
 39:            /$89/$46/$16           { mov   [bp+16],ax  }
 40:            /$89/$5e/$0e           { mov   [bp+0e],bx  }
 41:            /$89/$4e/$0c           { mov   [bp+0c],cx  }
 42:            /$89/$56/$0a           { mov   [bp+0a],dx  }
 43:            /$89/$76/$08           { mov   [bp+08],si  }
 44:            /$89/$7e/$06           { mov   [bp+06],di  }
 45:            /$8c/$46/$02);         { mov   [bp+02],es  }
 46:
 47:
 48: PROCEDURE call_int(vektor:pointer);
 49: INLINE($8f/$06/vektor/            { Adresse vom Stack }
 50:        $8f/$06/vektor+2/          {        "          }
 51:        $9c/                       { Pushf             }
 52:        $fa/                       { cli               }
 53:        $ff/$1e/vektor);           { call far [vektor] }
 54:
 55: {$F+}
 56: PROCEDURE int_13(flags,cs,ip,ax,bx,cx,dx,si,di,ds,es,bp:word);
 57: INTERRUPT;
 58: BEGIN
 59:   IF lo(dx)=disk THEN BEGIN
 60:     d:=TRUE;
 61:     IF hi(ax)=$16 THEN dcs:=TRUE ELSE dcs:=FALSE;
 62:   END ELSE d:=FALSE;
 63:   IF hi(ax) IN fkt THEN err:=TRUE ELSE err:=FALSE;
 64:
 65:   get_regs;
 66:   call_int(alt_13);
 67:   set_regs;
 68:
 69:   IF d THEN BEGIN
 70:     IF dcs THEN diskstatus^:=diskstatus^ AND $DF;
 71:     IF err AND (hi(ax)<>0) THEN diskstatus^:=diskstatus^ OR $20;
 72:   END;
 73: END;
 74: {$F-}
 75:
 76: FUNCTION DCS_ok(drivenr:BYTE):BOOLEAN;
 77: (* Prüft, ob Drive Disk-Change-Signal unterstützt.      *)
 78: VAR tmp : BOOLEAN;
 79: BEGIN
 80:   tmp:=FALSE;
 81:   WITH regs DO BEGIN
 82:     ah:=$15; dl:=drivenr;
 83:     intr($13,regs);
 84: (* AH=2, wenn Drive Disk-Change-Signal unterstützt      *)
 85:     IF ah=2 THEN tmp:=TRUE;
 86:   END;
 87:   DCS_ok:=tmp;
 88: END;
 89:
 90: BEGIN
 91:   cancel:=FALSE;
 92:   (* CMOS-RAM Register 10h auslesen:        *)
 93:   port[$70]:=$10;    (* Register anwählen *)
 94:   disks:=port[$71]; (* Inhalt nach disks *)
 95:   (* Laufwerkstypen feststellen:            *)
 96:   drivea:=(disks AND $F0)=$20;
 97:   driveb:=(disks AND $0F)=$02;
 98:   (* Disk-Change-Signal überprüfen          *)
 99:   IF drivea THEN drivea:=DCS_ok(0);
100:   IF driveb THEN driveb:=DCS_ok(1);
101:
102:   drive:='A'; disk:=0; diskstatus:=ptr($0040,$0090);
103:   disp:=ord(drivea)+ord(driveb)*2;
104:   CASE disp OF
105:   1: ;
106:   2: BEGIN
107:        drive:='B'; disk:=1; diskstatus:=ptr($0040,$0091);
108:      END;
109:   3: IF paramcount>0 THEN BEGIN
110:        pstr:=paramstr(1);
111:        drive:=upcase(pstr[1]);
112:        IF drive='B' THEN BEGIN
113:          diskstatus:=ptr($0040,$0091); disk:=1
114:        END
115:      END;
116:   ELSE cancel:=TRUE END;
117:
118:   writeln; writeln(
119:   'High-Capacity-Disk-Observer for 720 KByte (Ver. ',ver,')');
120:   writeln('(C) Ch. Hanisch, 1990');
121:   writeln; writeln(
122:   'Call: AT720 [<drive, if two 5.25"-HC-Drives are present>]');
123:   writeln;
124:   IF cancel THEN BEGIN
125:     writeln(
126:     'There is no 5.25-HC-Drive present or '+
127:     'Disk-Change-Signal is not supported!'#13#10'Canceled ...');
128:     Halt(255);
129:   END;
130:   getIntVec($13,alt_13);
131:   setIntVec($13,@int_13);
132:   writeln('successfully installed on drive ',drive,' ... ');
133:   keep(0);
134: END.
```

Bild 1** **Das Programm AT720 gestattet das Lesen und Schreiben von 720-KByte-Disketten in 1.2-MByte-Laufwerken des AT

Literatur:

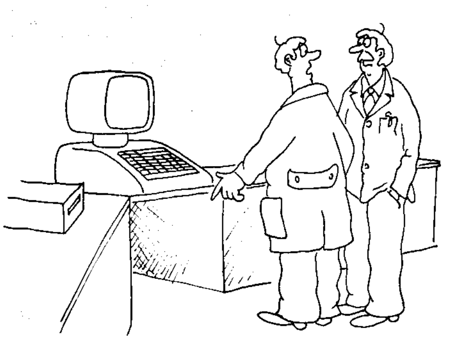
/1/ Ernst, M; Grell, D.: Damit die Scheibe spurt. c't, Hannover (1987) 11, S. 216
/2/ Bührmann, V.: Mit eigenem Format. c't Hannover (1990) 2, S. 202
/3/ Lorenz, W.: 360 KByte geschenkt. DOS-International, München (1989) 12, S. 86
/4/ Kirchhoff, H.: Auch Zwischengrößen lassen sich formatieren. Computer Persönlich, Haar bei München (1989) 26, S. 86
/5/ Weber, E.: Hohe Speicherkapazitäten mit einfachen Disketten. mc, München (1989) 6, S. 120

TERMINE

messtechnik in berlin

WER? Network GmbH
WANN? 14. bis 15. September 1990
WO? ICC Berlin, Hallen 14.2/15.2
WAS?
- Meßtechnik in Elektronik-Labor und in der Produktion
- industrielle und Kommunikationsmeßtechnik
- Mikrowellen und Optronik
- elektromagnetische Verträglichkeit

WIE? Informationen über Network GmbH, Wilhelm-Suhr-Straße 14, D-3055 Hagenburg



„Das Programm muß der Chef eingegeben haben, es ändert täglich seine Meinung."

Zeichnung: Steger

Einführung in MS-DOS

Teil 4

Wolfram Schulze, Uwe Schulze, Berlin

Was spielt sich in der Umgebung von Programmen ab? Umleitungen zum Nutzen des Programmierers? Filter für DOS? Antworten in diesem Kurs. Doch zunächst noch einmal zur Batch-Programmierung.

Wenn in einem Batch-Programm nur auf einen Tastendruck gewartet werden soll, ohne daß weiter ausgewertet wird, so verwendet man am einfachsten den Befehl **PAUSE**:
pause Bitte Diskette in a: einlegen
Die Meldung wird als Kommentar interpretiert und auf den Bildschirm ausgegeben, sofern ECHO ON geschaltet ist. DOS fordert mit **Weiter → eine Taste betaetigen** zur Weiterarbeit auf.
Mit **GOTO** lassen sich zwar Wiederholschleifen anlegen, aber die Festlegung von Abbruchbedingungen und die Übergabe anderer Parameter für jeden Umlauf gestalten sich umständlich. Deshalb stellt DOS die **FOR-DO**-Schleife zur Verfügung. Um beispielsweise alle Pascal-Quelltexte (Erweiterung .PAS) mit dem Turbo-Pascal-Kommandozeilencompiler TPC zu übersetzen, ist folgende Anweisung nötig.
for %%f in (*.pas) do tpc %%f
Solange noch .PAS-Dateien gefunden werden, wird der Aufruf **TPC dateiname** ausgeführt. Zur Belegung mit dem Dateinamen wird die Variable %%f benutzt. Sie können aber auch jeden anderen Buchstaben mit zwei vorangestellten Prozentzeichen verwenden (im Gegensatz zu Parametern, die aus einem Prozentzeichen und einer Ziffer bestehen). Wird ein FOR-Befehl aber interaktiv eingegeben (also direkt auf der DOS-Kommandoebene und nicht in einer BAT-Datei), so ist nur ein Prozentzeichen zu schreiben.
Des weiteren kann der FOR-Befehl dazu verwendet werden, Aufgaben auszuführen, die allein mit den Jokern nicht gelöst werden können. Zum Beispiel das Anzeigen aller Dateien mit den Erweiterungen .TXT und .DOC. in einem Befehl. Hier ist er:
for %%f in (*.txt,*.doc) do type %%f
Oder das Kopieren aller Programmdateien (also Erweiterungen COM, EXE und BAT) nach A:
for %%f in (*.com,*.exe,*.bat) do copy %%f a:%%f
Das Schachteln von Kommandos ist ebenfalls kein Problem. Wenn nur die Dateien auf die Sicherungsdiskette in A: kopiert werden sollen, die dort noch nicht vorhanden sind, so geschieht das mit
for %%f in (*.*) do if not exist a:%%f copy %%f a:
In großen Stapeldateien ist es zweckmäßig, durch Kommentare spätere Änderungen zu erleichtern. Dafür ist REM (Remark) vorgesehen. Kommentarzeilen können an jeder beliebigen Stelle eingefügt werden; sie beeinflussen den Programmverlauf nicht und erscheinen auch nicht auf dem Bildschirm. Beispiel:
rem Batch nach Anregung aus der MP
In Stapeldateien können neben den DOS-Befehlen auch beliebige Programme, also auch andere Stapeldateien, aufgerufen werden. Der Ruf anderer Batch-Dateien hat aber einen unerwünschten Nebeneffekt: Es erfolgt keine Rückkehr in das rufende Programm. In der gewünschten Weise kann ein Aufruf weiterer BAT-Dateien also nur am Ende (als letzter Befehl) erfolgen. Das hängt mit der internen Organisation der Abarbeitung von Stapeldateien zusammen. Ab DOS-Version 3.30 steht ein neuer Befehl zum Aufruf von Stapeldateien als Unterprogramm zur Verfügung: **CALL**. Rufen Sie also aus einer BAT-Datei die Datei TEST.BAT mit
call test
auf. Und eine weitere Verbesserung gibt es ab Version 3.30. Mit einem vorangestellten @ kann die Ausgabe einer Kommandozeile auf den Bildschirm unterdrückt werden. Selbst wenn Sie als erstes echo off stellen erscheint eben dieser Befehl noch auf dem Bildschirm. Schreiben Sie deshalb
@echo off
Auch vor andere Befehle kann ein @ gesetzt werden, das dann jeweils nur für diese Zeile gilt.
Benötigen Sie vor Version 3.30 den Call-Befehl, so schreiben Sie statt dessen
command /c test
Damit wird der Kommandointerpreter geladen und die Batch-Datei als Parameter übergeben – allerdings ist mehr Speicherplatz nötig.
Batch-Dateien können stets mit <CTRL> <C> (<Strg> <C>) unterbrochen werden, auch wenn BREAK OFF eingestellt ist. Dann wird aber nicht der aktuelle Befehl erreicht, sondern erst vor dem nächsten Befehl die Ausschrift
Stapeldatei abbrechen (J/N)
ausgegeben. Und weil wir gerade beim Abbrechen sind: Gerade in Batch-Dateien kann es schnell zu Laufwerksfehlern kommen, zumeist, weil keine (formatierte) Diskette im Laufwerk liegt. Man erhält die Ausschrift
Fehler beim Schreiben in Laufwerk A
Abbrechen, Wiederholen, Fehler
Die Eingabe des Anfangsbuchstabens genügt. Was unterscheidet diese Möglichkeiten? Mit A für Abbrechen wird der aktuelle Befehl beendet. In den meisten Fällen ist dieser Variante der Vorzug zu geben, weil sie keinen Schaden anrichtet. Interessante Möglichkeiten werden mit F für Fehler geboten. Hier wird die Fehlernummer an das rufende Programm (in diesem Falle die Batch-Datei) übergeben. Dort können sie wie oben beschrieben mit ERRORLEVEL getestet und entsprechende Maßnahmen eingeleitet werden. Viel ist damit im Falle eines Laufwerksfehlers nicht erreicht, aber es könnte zum Beispiel eine detaillierte Fehlermeldung gebracht werden. Wiederholen (w) kann zum Beispiel verwendet werden, wenn die Diskette mit einem Schreibschutz versehen war und nach dem Entfernen weiter gearbeitet werden soll. Aber Vorsicht: Gefährlich kann es sein, die Diskette zu wechseln und dann W zu drücken. In diesem Falle können im Speicher befindliche Informationen der alten Diskette (Directory) geschrieben und die Informationen der neuen Diskette zerstört werden.
Beachten Sie bitte, daß Batch-Dateien nicht den Namen bereits vorhandener Befehle (z. B. DIR.BAT) erhalten dürfen. Bei Eingabe eines Kommandos prüft DOS zuerst, ob es sich um einen internen (und damit reservierten) Befehl handelt. In diesem Falle wird dieser aufgerufen. Auch wenn der Name eines externen Befehls oder eines beliebigen Programms genutzt wird, kommt die BAT-Datei nicht zum Zuge. DOS sucht stets zuerst nach einer COM-Datei, dann nach EXE und erst ganz zum Schluß nach einer eventuellen BAT-Datei.
Da Sie nun schon erfahrene DOS-Programmierer sind, zum Schluß noch ein Trick zum Nachdenken. Wenn Sie einmal Zeit und Datum einer Datei ändern wollen, so ist das recht müßig. Sie müßte in einen Editor geladen und wieder gespeichert werden, damit DOS automatisch die neue Zeit setzt. Und was machen Sie mit einer Datei, die nicht aus Text besteht? Mit dem Befehl
copy /b datei +,,
setzen Sie Datum und Zeit einer oder mehrerer Dateien auf den aktuellen Stand. Der Schalter /b ist für eben diese Nicht-Textdateien zuständig, die sonst erheblich zusammengekürzt werden, weil das Dateiende falsch interpretiert wird. Probieren Sie doch mal. Ein bißchen mehr über Batch-Programmierung können Sie auch in MP 3/89, Seite 171, lesen.

Die Programmumgebung

Um gewisse Rahmenbedingungen für Programme festschreiben zu können, gehört zu jedem Programm ein kleiner Speicherbereich, in dem globale Parameter eingetragen sind. Er wird Programmumgebung (englisch Environment) genannt. Auch wenn noch kein Programm gestartet wurde, so ist zumindest der Befehlsinterpreter COMMAND.COM aktiv, und selbstverständlich besitzt auch er eine Programmumgebung. Einen Blick in das Environment gestattet der SET-Befehl. Schreiben Sie einfach
set
und es wird die aktuelle Programmumgebung angezeigt, die etwa so aussehen könnte:
COMSPEC=C:\COMMAND.COM
PATH=C:\DOS;C:\TOOLS
PROMPT=pg
Diese drei Befehle sind auch die einzigen, die ohne den SET-Befehl in die Programmumgebung eingetragen werden.
Vielleicht steht auch noch etwas mehr drin, je

nachdem, was Sie – wahrscheinlich in der AUTOEXEC.BAT – eingestellt haben. Sind vom Nutzer keine Eintragungen vorgenommen worden, so schreibt DOS zumindest diese beiden Zeilen ein:

PATH=
COMSPEC=C:\COMMAND.COM

Der Pfad ist also leer und als Befehlsinterpreter wird COMMAND.COM aus dem Hauptverzeichnis des Bootlaufwerks geholt. Es kann durchaus sinnvoll sein, diese Einstellung zu ändern, vor allem, wenn nicht von der Festplatte gebootet wurde. Der Kommandointerpreter COMMAND.COM besteht aus zwei Teilen, einem stets im Speicher verbleibenden (residenten) Teil und einem sogenannten transienten Teil, der von anderen Programmen überschrieben werden kann, da er während der Abarbeitung anderer Programme nicht benötigt wird. So kann Speicherplatz gespart werden. Nach dem Beenden eines Programms muß DOS stets diesen Teil des Kommandointerpreters nachladen, und zwar vom Startlaufwerk. Wurde von A: gestartet, so ist ein oftmaliger Diskettenwechsel nötig. Mit COMSPEC kann deshalb eingestellt werden, wo nach COMMAND-.COM gesucht werden soll. Günstig ist es, die Festplatte oder aber eine RAM-Diskette anzugeben. Die Verwendung einer RAM-Disk bringt außerdem Geschwindigkeitsvorteile und kann deshalb auch empfohlen werden, wenn von der Platte gebootet wurde. Dazu ist in der AUTOEXEC.BAT der Kommandointerpreter auf die RAM-Disk zu kopieren und COMSPEC einzustellen, z. B.

copy command.com e:
set comspec=e:\command.com

Zusätzlich zu den Systemvariablen kann der Nutzer beliebige eigene Variablen belegen; wiederum mit dem SET-Befehl, beispielsweise

set tag=montag oder
set name=pit oder
set codewort=roger

Der Variablen *tag* wird die Zeichenkette *montag* zugeordnet. Die erfolgreiche Eintragung kann man sich anschließend mit

set

ansehen. Dabei werden Sie sehen, daß DOS die Variablennamen stets in Großbuchstaben umwandelt, für die Belegung aber Groß- und Kleinschreibung unterscheidet, wie bei der Eingabe festgelegt.

Jedes Programm erbt das Environment von dem Programm, von dem es gestartet wird, also im allgemeinen von COMMAND.COM. Beim Beenden wird es wieder freigegeben, und das übergeordnete Environment wird aktiviert. Das hat zur Folge, daß alle inzwischen vorgenommenen Eintragungen verloren gehen, eine Tatsache, die häufig zu Fehlern führt. Von dieser Regelung sind beispielsweise SET-Befehle in Batch-Dateien betroffen. Da eine BAT-Datei beim Start eine eigene Programmumgebung erhält, werden alle Eintragungen dort vorgenommen. Nach dem Beenden befinden Sie sich wieder in der Programmumgebung des Kommandointerpreters, wo diese Einträge nicht mehr existieren. Positiv gesehen brauchen in Batch-Dateien gesetzte Systemvariablen am Ende also nicht gelöscht zu werden.

Bestes Beispiel für das Verschwinden von Systemvariablen ist die Arbeit mit dem Norton-Commander. Nutzen Sie den SET-Befehl bei geladenem Norton-Commander, so gelangen die Eintragungen nie in das Environment von COMMAND.COM oder in das des Norton-Commanders. Sie existieren nur unmittelbar während der Abarbeitung des SET-Befehls. Das hängt damit zusammen, daß bei jeder Befehlsausführung eine weitere COMMAND.COM-Kopie geladen und mit eigenem Environment versehen wird. Also Vorsicht.

An einem kleinem Beispiel kann man sich das Vererbungskonzept des Environments verdeutlichen. Schreiben Sie in eine BAT-Datei

set test=gesetzt
set

Beim Start wird die Variable TEST korrekt angezeigt. Wenn Sie aber mit *set* noch einmal nachsehen, so ist die Variable nicht mehr vorhanden. Sie existierte nur während der Zeit der Abarbeitung des Programms in seiner Programmumgebung.

Um die Nutzung der so eingetragenen Informationen in einem Programm muß sich der Nutzer selbst kümmern, denn weder das Betriebssystem noch Anwenderprogramme kennen die Variablen TAG, NAME oder CODEWORT. In den meisten Programmiersprachen existieren Konstrukte zum Abfragen von Environment-Variablen (GETENV in C, ENVSTR ab Turbo-Pascal 5.0), leider gibt es aber keine legale Möglichkeit, das Environment zu verändern. Das hängt damit zusammen, daß durch den Vererbungsmechanismus diese Eintragungen zum Programmende sowieso verlorengehen.

Einfacher als in Programmiersprachen lassen sich Informationen aus der Programmumgebung in Batch-Dateien nutzen. Dort wird der Variablenname einfach in Prozentzeichen eingeschlossen. Der Test auf das gesetzte Codewort kann in einer Batch-Datei so aussehen:

if not %codewort% == roger echo Unberechtigter Zugriff

Oder die Nutzung des Tages mit

echo Heute ist %tag%

Sollen Variablen stets präsent sei, so können diese SET-Befehle in die AUTOEXEC.BAT eingetragen werden.

Ein Entfernen aus dem Environment erfolgt durch Belegung mit einer leeren Zeichenkette – zum Löschen des TAG-Eintrages also

set tag=

Den Erfolg kann man sich mit

set

ansehen. Einige Programme kennen auch vordefinierte Variablen, die der Nutzer zur Einstellung von Standards belegen kann. Bei Compilern stehen häufig die Variablen INCLUDE (für den Suchpfad der Include-Dateien) und LIB (für den Suchpfad der Bibliotheken) zur Verfügung. Durch Anweisungen wie

set include=c:\compiler\inc
set lib=c:\compiler\bibl

kann dem Compiler mitgeteilt werden, wo er nach den entsprechenden Dateien zu suchen hat, ohne das bei jedem Aufruf angeben zu müssen.

Ein gestartetes Programm erhält ein eigenes Environment, und zwar als Kopie vom startenden Programm. Wenn also dBase aufgerufen wird, so erhält es eine Kopie der Programmumgebung von COMMAND.COM. So gehen keine Informationen verloren. Arbeiten mehrere Programme gleichzeitig (Norton-Commander, SideKick etc.), so verwaltet jedes Programm sein eigenes Environment. Zu beachten ist dabei, daß seine Größe beschränkt ist. Standardmäßig ist es 160 Byte groß. Probieren Sie doch einmal aus, einige sehr lange Parameter einzutragen:

set x=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx...usw.

Beim Ansehen mit *set* werden Sie feststellen, daß der Rest abgeschnitten wurde. In neueren Betriebssystemversionen wird gar eine Fehlermeldung ausgegeben, wenn die Programmumgebung voll ist. Da dem Environment eine wachsende Bedeutung beigemessen wird, ist ab DOS-Version 3.10 die Größe einstellbar (bis maximal 32 KByte). Das Einstellen kann immer geschehen, wenn COMMAND.COM geladen wird (siehe Folge 1), zum Beispiel

command /e:512

zur Festlegung von 512 Byte Environment. Soll diese Einstellung stets beim Start Bestand haben, so haben Sie

shell=command.com /e:512 \p

in die CONFIG.SYS einzutragen. Ein etwas größeres Environment ist bei vielen neueren Programmen nötig; gehen Sie aber auch nicht zu großzügig mit dem Speicherplatz um, es ist zu bedenken, daß jedes nachfolgend gestartete Programm eine ebenso große Kopie erhält.

Mit dem Shell-Befehl ist es auch möglich, ein anderes Programm als COMMAND.COM zur Kommandooberfläche zu erklären, etwa ein selbstgeschriebenes. Es funktioniert auch, den Norton-Commander dort festzulegen, aber alle Systembefehle (DIR,...) stehen nicht mehr zur Verfügung, weil sie nur von Norton an COMMAND.COM weitergegeben werden, und verlassen kann man den Commander auch nicht, weil kein Kommandointerpreter „darunterliegt".

Umleitung von Ein- und Ausgabe

Bis jetzt wurde bei der Bedienung ganz selbstverständlich vorausgesetzt, daß alle Eingaben von der Tastatur kommen und alle Ausgaben auf dem Bildschirm erscheinen. Warum eigentlich? Es gibt doch noch andere Möglichkeiten: Drucker, serielle Schnittstellen und Dateien. DOS betrachtet Bildschirm und

SET	–	Anzeigen des Environments
SET variable=string	–	Eintragung einer Variablen in das Environment
SET variable=	–	Löschen einer Eintragung

Bild 1

Tastatur (zusammen Konsole) als Standard für Ein- und Ausgabe. In Beschreibungen findet man deshalb auch STDIN (für Standard Input) und STDOUT (für Standard Output). Um eine möglichst flexible Handhabung der Ein- und Ausgabe zu erreichen, wurden ab der Version 2.00 Methoden zur Umleitung (englisch Redirection) von Ein- und Ausgaben aufgenommen. So können mit dem Kleinerzeichen (<) die Herkunft der Eingaben und mit dem Größerzeichen (>) das Ziel der Ausgaben festgelegt werden. Man spricht auch von der Steuerung eines Datenstromes. In Teil 1 (in MP 4/90) sind im Zusammenhang mit dem COPY-Befehl die DOS-Geräte bereits vorgestellt worden:
PRN oder LPT1 für den Drucker, COM1 oder AUX für die serielle Schnittstelle sowie NUL als „Papierkorb". Um also das aktuelle Verzeichnis nicht auf den Bildschirm (wie gewohnt), sondern auf den Drucker auszugeben (Einschalten nicht vergessen!), schreiben Sie

dir > prn

Mit <CTRL> <P> (<Strg> <P>) geht's auch – trotzdem eine wertvolle Möglichkeit. PRN und LPT1 sowie COM1 und AUX können synonym verwendet werden.
Das Gerät NUL ist ein Scheingerät, das die umgeleiteten Daten ganz einfach verschwinden läßt. Nicht besonders wichtig, aber auch nicht sinnlos. Man kann sich damit von unliebsamen Ausschriften befreien; nützlich in langen und stets wieder verwendeten Batch-Dateien. Wenn Sie alle Dateien des Tageswerkes mit

copy *.* a: > nul

auf eine Sicherungsdiskette retten, so entfällt die Ausschrift

xxx Datei(en) kopiert

Eine Besonderheit von DOS ist, daß eine völlige Gleichbehandlung von Geräten (Konsole, Drucker) und Dateien erfolgt. An jeder Stelle, wo ein Gerät angegeben wird, kann auch eine Datei stehen. Sie können Ihr Inhaltsverzeichnis also auch in eine Datei schreiben. Versuchen Sie es mal mit

dir > dir.dat

Es wird eine Datei DIR.DAT angelegt, in der das steht, was sonst auf den Bildschirm kommt. Schauen Sie doch mal rein – mit einem Texteditor oder mit

type dir.dat

Auf diese Weise habe ich beispielsweise die Originalausschriften der Programme aufgefangen, um Sie im Kurs darzustellen. Für die Beschreibung von CHKDSK in Teil 2 (in MP 6/90) wurden die Ausschriften mit

chkdsk > text

in eine Datei geleitet und dann in den Kurs-Text kopiert.
Die Umleitung von Ein- und Ausgabe ist aber nicht auf DOS-Kommandos beschränkt; grundsätzlich gilt gleiches für beliebige Programme – vorausgesetzt ihre Ausgaben gehen über DOS. Sie werden aber feststellen, daß viele Programme eigene Routinen für die Ein- und Ausgabe verwenden. Angesichts Fenstertechnik und Mausbedienung: Tendenz steigend. Im Zweifelsfalle hilft nur probieren.
Die Umlenkung der Eingabe läßt sich nur selten sinnvoll anwenden. Eine der wenigen Anwendungen wurde in der letzten Folge beim Programm TASTE gezeigt.
Ist nicht beabsichtigt, bei der Ausgabeumleitung eine neue Datei anzulegen, so kann auch der Anfügeoperator (zwei Größerzeichen) eingesetzt werden. Beispiel zum Schreiben der Verzeichnisse aller Laufwerke in eine Datei:

dir a: > inhalt
dir b: >> inhalt
dir c: >> inhalt

Der erste Befehl schreibt die Ausgaben in eine Datei, die neu angelegt wird. Die weiteren Kommandos fügen ihre Ausgaben an diese Datei an, die damit immer länger wird. Die vorherigen Eintragungen gehen nicht verloren. Das Verketten von Dateien kann ja auch mit COPY und + vorgenommen werden, aber das Anfügen von Ausgaben eines Programms wird nur so erreicht. Die bis jetzt kennengelernten vielfältigen Möglichkeiten erlauben es auch, die gleiche Aktion mit verschiedenen Befehlen zu erreichen. So erfolgt die Ausgabe einer Datei auf den Drucker sowohl mit

copy datei prn als auch mit
type datei > prn

Ist der gleiche Effekt auch mit

copy datei con > prn

zu erreichen? Sie werden sehen, daß die Datei selbst auf dem Bildschirm erscheint und nur die Meldung des COPY-Befehls

(xxx Datei(en) kopiert)

auf dem Drucker. Denn die Ausgaben des COPY-Befehls (und nur die sind umzulenken) bestehen aus den jeweiligen Ausschriften – nicht aus dem Ziel des Kopiervorganges, der eine interne Aktion des Befehls ist. So würde

copy datei > prn

auch nur die Fehlerausschrift „Datei kann nicht auf sich selbst kopiert werden" auf den Drucker bringen.
Probieren Sie am besten noch ein wenig.

Filter

Über die Umleitung von Ein- und Ausgabe hinaus kann der Datenstrom auch durch ein anderes Programm hindurchgeleitet werden. Dazu werden die Ausgaben des einen Programms einfach als Eingaben des anderen betrachtet. Diese Programme heißen Filter (englisch Pipes) und werden mit dem Ziel benutzt, die Ausgabe in irgendeiner Weise zu verändern, eben zu filtern. Zum Lieferumfang von DOS gehören drei Filter: SORT, FIND und MORE. Darüber hinaus kann man sich aber auch selbst Filterprogramme schreiben. Am einfachsten zu gebrauchen ist MORE. Es gibt stets nur 23 Zeilen auf den Bildschirm aus und wartet dann auf einen Tastendruck. Zum Anzeigen eines langen Directories verwendet man

dir | more

Wie zu sehen, dient zur Filterung eines Datenstromes der senkrechte Strich (|). Wenn Sie damit Probleme auf der Tastatur haben, so erinnern Sie sich bitte der Nutzung des ASCII-Codes. Mit dem Wissen, daß der Code für den Strich 124 (dezimal) ist, können Sie ihn erzeugen, indem die ALT-Taste gedrückt, rechts 124 getippt und die ALT-Taste wieder losgelassen wird. Mit dem Filterzeichen werden die Ausgaben des DIR-Befehls als Eingaben von MORE weitergeleitet. Das MORE-Filter erlaubt so eine seitenweise Ausgabe. Noch weniger als für den DIR-Befehl (wo man ja den /p-Schalter hat) ist MORE für TYPE:

type datei | more

Das SORT-Filter sortiert den Datenstrom. Zur alphabetisch sortierten Ausgabe des Verzeichnisses schreiben Sie

dir | sort

Es geht auch absteigend mit

dir | sort /r

Es muß nicht unbedingt nach dem Anfangsbuchstaben sortiert werden; die Anweisung

dir | sort /+9

sortiert die Dateien nach dem Typ, der gerade in der 9. Spalte einer jeden Zeile steht. Beachten Sie bitte, daß damit nur die Anzeige auf dem Bildschirm geändert wird, nicht etwa die Reihenfolge der Einträge in das Inhaltsverzeichnis.
Durch das SORT-Filter können auch Dateien geschickt werden. Mit

sort < datei

erfolgt eine sortierte Ausgabe auf den Bildschirm, und mit

sort < altdatei > neudatei

wird das Ergebnis wiederum in eine Datei geschrieben. Probieren Sie es mal mit einer beliebigen Textdatei.
Das Sortieren der oben angelegten Datei INHALT mit den Verzeichnissen der Laufwerke A:, B: und C: würde diese Verzeichnisse übrigens bunt durcheinandermischen. Zum Schluß überlegen Sie bitte einmal, warum bei den letzten Befehlen das Filter am Anfang der Kommandozeile stand und nicht einfach **dir > sort** geschrieben werden kann. Sie können es gefahrlos ausprobieren: Es wird eine Datei mit Namen SORT angelegt, und das war wohl nicht Sinn der Sache.
Noch vielfältiger kann das FIND-Filter eingesetzt werden, das ungefähr das macht, was man von einem Filter erwartet: Es filtert bestimmte Teile heraus. Gearbeitet wird zeilenweise, das heißt, es wird die gesamte Zeile ausgegeben, die die Suchbedingung erfüllt. So werden mit

dir | find "BAT"

nur die Dateien angezeigt, die die Zeichenkette "BAT" enthalten. Nicht eben genial, das hätte man auch mit Jokern erreicht. Aber können Sie auch alle Dateien außer den BAT-Dateien anzeigen?
Kein Problem:

dir | find /v "BAT"

Achten Sie bitte darauf, daß an dieser Stelle Großbuchstaben für die Zeichenkette zu verwenden sind, da die Eintragungen im Directory sämtlich in Großbuchstaben umgewandelt werden. Ähnlich müssen Sie verfahren, wenn Sie nur die Unterverzeichnisse anzeigen wollen:

dir | find "<"

Oder das Anzeigen aller Dateien, die im September erstellt wurden:

dir | find ".09."

Wichtig ist nur, eine signifikante Zeichenkette zu finden, die die entsprechenden Zeilen von allen anderen unterscheidet.
Statt als Filter kann man FIND auch als eigenständiges Programm verwenden, etwa

um die Zeilen dieses Kurses zu zählen, in denen das Wort „DOS“ vorkommt:

find /c "DOS" dos4.doc

Die Antwort ist

---------- **dos4.doc: 14**

Bis jetzt kam also in 14 Zeilen das Wort „DOS“ vor, was nicht unbedingt bedeutet, daß es nur 14mal im Text Verwendung fand, da es ja auch mehrmals auf einer Zeile stehen kann und dann nur einmal gezählt wird. Es wird – wie schon erwähnt – zeilenweise gearbeitet.

```
                    F I L T E R

Seitenweise Ausgabe

MORE

Sortieren

SORT [/R] [/+n]

/R     umgekehrte Sortierreihenfolge
/+n    sortieren nach Spalte n

Suchen nach Zeichenketten

FIND [/C] [/N] [/V] "string"

/C      Zählen der Zeilen, die die
        Zeichenkette enthalten
/N      Anzeige mit Zeilennummer
/V      Suchen aller Zeilen, die die
        Zeichenkette nicht enthalten
```

Bild 2

Durch Leerzeichen getrennt können auch mehrere Dateien aufgeführt werden, die dann nacheinander durchmustert werden; der Einsatz von Jokern für Dateiengruppen ist aber nicht erlaubt.

Niemand hindert Sie, ihre Daten durch beliebig viele Filter zu schicken, beispielsweise um alle Dateien, die nicht die Erweiterung DOC haben, nach Namen sortiert seitenweise auf den Bildschirm auszugeben:

dir | find /v "DOC" | sort | more

Sie werden bemerken, daß die Ausführung etwas länger dauert, denn DOS baut intern Hilfsdateien auf, in denen gesucht und sortiert wird. Diese temporären Dateien werden automatisch gelöscht, können aber unter Umständen bei einem Absturz oder einem Reset während der Abarbeitung auf der Festplatte zurückbleiben. Der Name dieser Dateien mutet seltsam an (z. B. 0e14081a), weil er bei neueren Betriebssystemversionen aus der Zeit gebildet wird, um im Netzbetrieb nicht zu kollidieren.

Denken Sie nur nicht, Filter seien eine komplizierte Sache. Jeder kann sich mit einfachen Mitteln selbst welche schreiben. Diese Programme müssen den Datenstrom von der Standardeingabe einlesen, ihre Aufgabe ausführen und wieder zur Standardausgabe schreiben. Sehen sie, wie kurz ein Pascal-Programm ist, das – als Filter verwendet – Texte mit Zeilennummern versieht:

```
program number;
var zeile : string;
  i     : integer;
begin
  i:=1;
  while not eof do begin
  readln(zeile);
  writeln(i:3,'      ',zeile);
  i:=i+1;
  end;
end.
```

Durch dieses Filter können jetzt beliebige Texte geschickt werden, z. B. das Inhaltsverzeichnis mit

dir | number

oder auch dieser Kurs:

type dos4.doc | number

Es ist auch kein Problem den Text mit Zeilennummern wieder abzuspeichern:

number < dos4.doc > dos4num.doc

Mit Hilfe des Kleiner- und Größerzeichens wird der Ein- und Ausgabestrom durch das Filter geleitet. Unter dem Namen „dos4num-.doc“ steht der numerierte Text wieder auf der Platte. Auf diese Art und Weise können Sie sich ganz leicht beliebige Filter selbst schreiben. Weitergehende Informationen zu Filtern finden sich in MP 9/89, S. 263.

Filter können weiterhin dazu verwendet werden, Rückfragen von Befehlen zu begegnen. Wenn am Ende einer Sitzung nach dem Sichern stets alle Dateien gelöscht werden sollen, so erzeugt der Befehl "**del *.***" stets die Ausschrift "**Wirklich löschen? j/n**", was in einer BAT-Datei lästig sein kann. Mit

echo j | del *.*

wird nicht angehalten. Etwas schwerer ist es schon, Befehle zu bedienen, die nur ein <ENTER> erwarten, zum Beispiel FORMAT. Der Befehl

echo.| format a:

beginnt mit dem Formatieren, ohne auf Tastendruck zu warten. Zwischen Punkt und senkrechtem Strich darf kein Leerzeichen stehen. Auch DATE und TIME warten stets auf das Drücken der ENTER-Taste, wenn Datum oder Zeit nur angezeigt werden sollen. Kein Problem mit

echo.| time

Und wenn nur die Zeile mit der Zeit gefragt ist, so filtert man den Rest einfach weg:

echo.| time | find "."

Natürlich gibt das nur in Batchdateien richtig Sinn. Ein <ENTER> kann übrigens auch mit dem MORE-Filter erzeugt werden. Damit wird ausgenutzt, daß es stets eine Leerzeile einfügt. Statt des oben beschriebenen Befehls kann so auch

echo | more | time

geschrieben werden.

Wie Sie sehen, muß man manchmal schon ganz schöne Klimmzüge veranstalten, um elegante Lösungen zu erreichen, aber die DOS-Kommandosprache ist halt recht klein – dafür aber leicht zu erlernen.

Ihnen wird aufgefallen sein, daß immer häufiger von Dienstprogrammen die Rede ist. Diese haben heute die meisten DOS-Befehle verdrängt, die hier – um ein solides Grundwissen zu vermitteln – ausführlich beschrieben wurden. In der nächsten Folge soll deshalb ein Programm im Mittelpunkt stehen, das sich – ob als Dienstprogramm oder als Nutzeroberfläche – beachtlicher Beliebtheit erfreut: der Norton-Commander.

Neuerungen von MS-DOS 4.x

Obwohl die neueste MS-DOS-Version die Nummer 4.x trägt, wird in der Praxis überwiegend noch mit Version 3.3 gearbeitet. Was bietet ein Umstieg für Vorteile? Auffallendste äußere Neuerung ist die zum Betriebssystem gehörige grafische Benutzeroberfläche (DOS-Shell genannt), die aber – verglichen mit Windows der GEM – recht primitiv wirkt und sich auch mit beliebten Shells auf alphanumerischer Basis (Norton Commander, PC Tools) nicht messen kann.

Die wirklich gravierende Neuerung besteht in der Nutzung von Festplatten mit mehr als 32 MByte als ein logisches Laufwerk. Unter DOS 3.x ist eine größere Festplatte in mehrere Laufwerke aufzuteilen (zu partitionieren). Angesichts ständig billiger werdender großer Festplatten mag dieser Umstand ein echter Beweggrund für einen Umstieg auf eine neue Version von MS-DOS sein – allerdings um den Preis, daß einige hardwarenahe Programme nicht mehr korrekt arbeiten. Probleme wurden mit den Norton Utilities und leider auch mit dem neuen Wordperfekt 5.1 festgestellt. Ursache dafür ist die Erweiterung einer DOS-internen Tabelle, des DOS-Parameter-Blocks (DPB), der die nähere Beschreibung der Datenträger enthält. Die alte Beschränkung auf 32 MByte ergibt sich übrigens aus der Sektoradressierung mit einer 16-Bit-Zahl. 65 536 Sektoren zu je 512 Byte sind so erreichbar, was einer Kapazität von 32 MByte entspricht. Von untergeordneter Bedeutung dürfte die Unterstützung des Expanded Memory durch DOS 4.x sein, da die für den 8086 eingeführte Hardware auf ATs kaum genutzt wird. Mit dem Treiber XMA2EMS-.SYS läßt sich EMS als Datenbereich oder Cachepuffer einrichten (beispielsweise läßt sich der mit *Buffers* reservierte Speicherbereich auslagern), VDISK.SYS unterstützt eine RAM-Disk im EMS. Wichtiger noch mag der Treiber XMAEM.SYS sein, der EMS im Extended Memory (wie ihn viele ATs besitzen) emuliert.

Vier Befehle sind bei DOS 4.x neu hinzugekommen:

- DOSSHELL – Start der grafischen Nutzeroberfläche
- INSTALL – Installation von DOS-Programmen bereits in Config-.SYS
- MEM – Zeigt die Hauptspeicherbelegung detailliert an, wozu bisher Hilfsprogramme (Mapmem) notwendig waren.
- SWITCHES – Schaltet die erweiterten Tastaturfunktionen ab (einige Programme erfordern das).

Zusätzlich wurde der Funktionsumfang einer Reihe vorhandener Befehle erweitert. Beispielsweise zeigt *Tree* das Verzeichnissystem jetzt wirklich als Baum.

Übrigens: Für Ende dieses Jahres hat Microsoft – entgegen ursprünglichen Absichten – eine grundlegend neue Version von MS-DOS angekündigt, die voraussichtlich die Nummer 5.0 tragen wird.

Zwei A 7150 an einem Drucker

Jürgen Lang, Lugau

Das bedarfsweise Umstecken des Anschlußkabels ist sicher keine Dauerlösung, wenn man einen Drucker wahlweise an zwei PCs betreiben will. Im folgenden wird deshalb eine programmtechnische Lösung für das DOS-kompatible Betriebssystem DCP in Form zweier kleiner Hintergrundprogramme vorgestellt, die die auszudruckenden Daten über die V.24-Schnittstelle an den Partnerrechner und von dort über die Centronics-Schnittstelle an den Drucker weiterleiten.

```
Program LPTSend;

{Programmiersprache: Turbo-Pascal ab 4.0
 Autor: J.Lang                                  }

{$M 1024,0,0}

uses CRT,DOS;

Const   Data    = $0318;
        Control = $031C;
        CTC     = $0312;

var     alt_17  :  Pointer;
        bvar    :  Byte;

Procedure V24Out(Kanal:Word; Wert:Byte);

Begin
  Port[Kanal]:=Wert
End;

Function V24In(Kanal:Word):Byte;

Begin
  V24In:=Port[Kanal];
End;

Procedure V24Init;

Begin
  V24Out(Control,$00);   {Schreibregister 0}
  V24Out(Control,$18);   {SIO-Kanal zurücksetzen}
  V24Out(CTC,$03);       {CTC-Kanal zurücksetzen}
  V24Out(CTC,$07);       {Vorteiler=16 Zeitkonstante folgt}
  V24Out(CTC,$01);       {Timekonstante}
  V24Out(Control,$04);   {Schreibregister 4}
  V24Out(Control,$4D);   {Teilerfaktor = 16}
  V24Out(Control,$03);   {Schreibregister 3}
  V24Out(Control,$C1);
  V24Out(Control,$05);   {Schreibregister 5}
  V24Out(Control,$E8);   {19200 bd,8-bit,2 Stopp,ungerade Par.,
                          Tx/Rx frei}
End;

Procedure SendeByte(SByte:Byte);

Var  Temp : Byte;

Begin
  V24Out(Data,SByte);
  Repeat
    V24Out(Control,$00);
  Until (V24In(Control) and $01) = $01;
  Temp:=V24In(Data);
End;

procedure neu_17(Flags,CS,IP,AX,BX,CX,DX,SI,DI,DS,ES,BP:Word);
interrupt;

begin
  inline($FA);
  case Hi(ax) of
    0 : begin
          bvar:=Lo(ax);

          SendeByte(bvar);
          ax:=$9000;
        end;
    1 : ax:=$9000;
    2 : ax:=$9000;
  end;
  inline($FB);
end;

Begin
  GetIntVec($17,alt_17);
  SetIntVec($17,@neu_17);
  V24Init;
  WriteLn('LPTSend ist installiert');
  Keep(0);
End.
```

Bild 1

Hardwaretechnische Voraussetzungen

Beide zu koppelnden V.24-Schnittstellen müssen auf asynchrone Datenübertragung (beim A 7150 mittels DIL-Schaltern auf der ASP-Karte) eingestellt und mit einem Nullmodem-Kabel, das mindestens folgende Verbindungen herstellt, miteinander verbunden werden:

Kabelende a				Kabelende b
Schutzerde	1	———	1	Schutzerde
TxD	2	———	3	RxD
RxD	3	———	2	TxD
Signalerde	7	———	7	Signalerde

Programmbeschreibung

Die vorgestellten Programme sind in Turbo-Pascal geschrieben und ab Version 4.0 com-

```
Program LPTRec;

{Programmiersprache: Turbo-Pascal ab 4.0
 Autor: J.Lang                                  }

{$M 1024,0,0}
{$S-}

uses DOS,CRT;

Const   Data                    = $0318;
        Control                 = $031C;
        CTC                     = $0312;
        raster      : Integer   = 32;
        tickercount : Integer   = 0;
        EOI         : Byte      = $20;

Var     alt_8       : Pointer;

Procedure V24Out(Kanal:Word; Wert:Byte);

Begin
  Port[Kanal]:=Wert
End;

Function V24In(Kanal:Word):Byte;

Begin
  V24In:=Port[Kanal];
End;

Procedure V24Init;

Begin
  V24Out(Control,$00);   {Schreibregister 0}
  V24Out(Control,$18);   {SIO-Kanal zurücksetzen}
  V24Out(CTC,$03);       {CTC-Kanal zurücksetzen}
  V24Out(CTC,$07);       {Vorteiler=16 Zeitkonstante folgt}
  V24Out(CTC,$01);       {Timekonstante}
  V24Out(Control,$04);   {Schreibregister 4}
  V24Out(Control,$4D);   {Teilerfaktor = 16}
  V24Out(Control,$03);   {Schreibregister 3}
  V24Out(Control,$C1);
  V24Out(Control,$05);   {Schreibregister 5}
  V24Out(Control,$E8);   {19200 bd,8-bit,2 Stopp,ungerade Par.,
                          Tx/Rx frei}
End;

Function EmpfangByte:Byte;

Var  Temp : Byte;

Begin
  Repeat
    V24Out(Control,$00);
  Until (V24In(Control) and $01) = $01;
  Temp:=V24In(Data);
  V24Out(Data,Temp);
  EmpfangByte:=Temp;
End;

procedure Druck(x:Byte);

begin
  repeat until port[$C8]=$C6;

  port[$CA]:=x;
  port[$CE]:=$07;
  port[$CE]:=$06;
end;

procedure neu_8;
interrupt;

begin
  inline($FA);
  V24Out(Control,$00);
  if V24In(Control) and $01 = $01 then Druck(EmpfangByte);
  tickercount:=tickercount+raster;
  if tickercount>=256 then
    begin
      tickercount:=Lo(tickercount);
      inline($9C/$FF/$1E/alt_8);
    end
  else port[$C0]:=EOI;
  inline($FB);
end;

Begin
  GetIntVec($08,alt_8);
  SetIntVec($08,@neu_8);
  port[$D0]:=0;
  port[$d0]:=raster;
  V24Init;
  Writeln('LPTRec ist installiert');
  Keep(0);
End.
```

Bild 2

pilierbar. Wie alle TSR-Programme dürfen diese nur als compiliertes Programm von Diskette oder Festplatte gestartet werden.
Das Programm *LPTSend* übernimmt die Übertragung der auszudruckenden Daten an den Partnerrechner. Dazu wird durch das Ladeprogramm eine neue Behandlungsroutine für den Druckerinterrupt installiert.
Das Übertragungsprotokoll ist relativ einfach gehalten: Die Prozedur *SendeByte* sendet das betreffende Zeichen an den Partnerrechner und wartet auf eine Empfangsbestätigung in Form des vom Empfänger zurückgesendeten Zeichens. Im Falle der Direktkopplung wird damit eine fehlerfreie Übertragung gewährleistet.
Das Programm *LPTRec* fragt den V.24-Eingang im Polling-Betrieb ab und leitet die empfangenen Zeichen zur Centronics-Schnittstelle. Genutzt wird dazu der Timer-Interrupt 8, der pro Sekunde 18,2mal ausgelöst wird und normalerweise den Zähler für die Systemuhr weiterschaltet. Damit wäre eine maximale Druckgeschwindigkeit von 18 Zeichen pro Sekunde möglich. Zwecks Erhöhung der Druckgeschwindigkeit wird die Zeitkonstante im CTC auf 32 herabgesetzt und somit die Taktgeschwindigkeit um den Faktor 8 erhöht. Die neue Interrupt-Behandlungsroutine sichert das korrekte Weiterschalten der Systemuhr, jedenfalls solange der Drucker Bereitschaft meldet. Da die hier verwendete Druckprozedur auf die Druckbereitschaft wartet, können Interrupts ausgelassen werden, was ein „Nachgehen" der Systemuhr verursacht.

Datenbanksysteme im Vergleich

Gert Klein, Halle

Mit Datenbanksystemen können zirka 30 % aller mit einem PC zu lösenden Aufgaben bearbeitet werden. Deshalb verdienen Einsatzvorbereitung und Systemauswahl besondere Beachtung. Wichtig sind die Fragen: Welches Datenbanksystem soll man anwenden? Welche Bedingungen sind einzuhalten? Wie befinden sich Problemgrenzen mit Hardware- und Softwaregrenzen im Einklang? Dazu sollen in dem folgenden Beitrag Denkanstöße vermittelt werden. Eine umfassende Betrachtung aller Umstände ist jedoch aufgrund der Komplexitität der Problemstellung und der Kürze des Beitrages nicht möglich.

Vergleich der wichtigsten Datenbanksysteme

Vergleichsbasis

MS-DOS ist das am meisten verbreitete Betriebssystem. Die Kriterien für die wichtigsten Datenbanksysteme sind deshalb ihre Lauffähigkeit unter diesem Betriebssystem und ihre Häufigkeit im Einsatz (in der DDR). Damit ergab sich, daß folgende Systeme zu testen waren (Versionsangaben betreffen die getestete Software):

- dBase II — Version 2.41
- dBase III — Version 1.10
- dBase III Plus — Version 1.0
- dBase IV — Version 1.0
- Clipper '87 — Version 2.24
- Foxbase Plus 2.1 — Version 2.1

Der Test wurde an Hand der Standardversionen durchgeführt, das heißt, die unter Umständen vorhandenen Tools, die der Aufwertung dieser Softwarepakete dienen, wurden nur am Rand mitbewertet. Getestet wurden nur Vollversionen, das heißt, da zu Foxbase die neueste Version Foxpro nur als Demoversion vorlag, wurde hier nur eine Aussage zur Funktionalität getroffen (also keine Geschwindigkeitstests). Diese Datenbanksysteme erfüllen die Forderungen des Coddschen relationalen Datenmodells /1/ im übertragenen Sinne oder besser, sie sind quasirelational.
Als Hardwarebasis für den durchgeführten Test diente ein PC 1512 mit einem auf 640 KByte aufgerüsteten Hauptspeicher, mit Floppy-Laufwerken, mit einer Harddisk und mit dem Betriebssystem MS-DOS 3.2. Die Ausrüstung mit 640 KByte RAM ist nur bei dBase IV bindend, die anderen Softwarepakete sind auch mit 512 KByte RAM funktionsfähig.
Inhalt und Funktionalität dieser Datenbanksysteme werden als bekannt vorausgesetzt (weiterführende Literatur siehe Anhang).

Funktioneller Vergleich

dBase II als ältestes der verglichenen Datenbanksysteme war schon für 8-Bit-PCs verfügbar. Die 16-Bit-Variante von dBase II enthält keinerlei neue Möglichkeiten, das heißt, sie ist eine bloße Umsetzung. dBase III und sein jüngerer Bruder dBase III Plus sind dagegen echte Neuentwicklungen, die vor allem an die erweiterten Eigenschaften der 16-Bit-PCs angepaßt wurden.
Nach diesen Systemen aus der dBase-Familie machten vor allem die sogenannten Turbo-Versionen dieser Datenbanksysteme auf sich aufmerksam. Ihr augenscheinlichster Hauptvorteil – die hohe Geschwindigkeit. Danach betrat mit dBase IV noch ein weiteres Mitglied der dBase-Familie die Szene der EDV-Anwendung. Übertroffen wird dBase IV jedoch von Foxpro.
Die Bewertung der Möglichkeiten an Hand notwendiger Funktionen für jedes einzelne System ist aufwendig und deshalb nicht jedem Anwender zuzumuten. Daher wurde von mir ein praxisorientierter Vergleich unter Verwendung allgemeiner Testdaten (Bild 1) und eines auf jedem Einzelsystem lauffähigen Softwaremoduls (Bild 2) durchgeführt (siehe Tafeln 1 und 2). Dieser wurde nur für dBase II leicht modifiziert, da dort einige Befehle fehlen.

```
Datenbankstruktur        : C:bench.dbf
Anzahl der Datensaetze :       0
Letztes Aenderungsdatum: 12.06.90
Feld   Feldname       Typ          Laenge    Dez
   1   BENCH          Numerisch       4
   2   TEXT           Zeichen       123
** Gesamt **                        128
```

***Bild 1** Struktur der Testdatenbank BENCH.DBF*

Die Auswahl erfolgte für wichtige und vorteilhafte Funktionen zur Darstellung des Leistungsumfanges und für Aussagen zur Kompatibilität.
Funktionen, die von den Systemen in etwa gleich gut erfüllt werden, stehen nicht im Vordergrund. Zu beachten ist auch, daß bei Erfüllung von Forderungen insgesamt Unterschiede im Detail auftreten können. Generell ist zu beachten, daß die erwähnten Tools zur Aufwertung oft nur für das in der jeweiligen Beschreibung angekündigte System lauffähig sind. Probleme, wie Speicheraufrüstung auf 640 KByte, existieren auch. Nachteilig ist ebenfalls generell, daß syntaktische Unterschiede kleinerer oder größerer Art bestehen (Bild 3) und daher die Entscheidung für nur ein System Vorteile bringt.
Die betrachteten Datenbanksysteme sind zumindest nicht 100prozentig kompatibel, bedingt durch syntaktische Unterschiede sowie durch das Fehlen von Befehlen und Funktionen. Gleiches gilt beispielsweise auch für den Komfort und das Vorhandensein eines Texteditors. dBase IV und Foxpro haben im Gegensatz zu Clipper für die Projektierung eine komplette Entwicklungsumgebung. Nachteilig ist bei dBase IV die Geschwindigkeit in der Dialogarbeit. Durch das Nachladen von Programmen entstehen Wartephasen

```
SET  TALK   OFF
CLEAR
@  1,0 SAY   'Benchmarktest am  '
?? DATE()
@  3,0 SAY   'Test läuft,bitte warten ...'
X=0
? 'Beginn des Zähltests: '
?? TIME()
DO  WHILE   X<10000
    X=X+1
ENDDO
? ' Ende des Zähltests: '+TIME()
WAIT
SET  SAFETY   OFF
USE  BENCH
ZAP
X=0
Y=1000
? 'Beginn des APPEND-Tests    : '
?? TIME()
DO  WHILE   X<1000
    APPEND    BLANK
    REPLACE    BENCH    WITH   Y
    Y=Y-1
    X=X+1
ENDDO
? ' Ende des APPEND-Tests     : '+TIME()
WAIT
USE  BENCH   INDEX  BENCH
? 'Beginn des Index-Tests : '
?? TIME()
REINDEX
? ' Ende des Index-Tests : '+TIME()
WAIT
X=0
? 'Beginn des Such-Tests  : '
?? TIME()
DO  WHILE   X<1000
    SEEK   999
    X=X+1
ENDDO
? ' Ende des Such-Tests  : '+TIME()
WAIT
GO  TOP
X=0
? 'Beginn des Ersetzen-Tests : '
?? TIME()
DO  WHILE   X<1000
    SEEK  X
    REPLACE    TEXT   WITH   '***'+;
    '*********************'+;
    '*********************'+;
    '*********************'+;
    '*********************'+;
    '*********************'+;
    '*********************'
    X=X+1
ENDDO
? ' Ende des Ersetzen-Tests : '
?? TIME()
?
? ' Ende der Benchmark-Tests'
?
WAIT
RETURN
```

***Bild 2** Programm zur Ermittlung der Laufzeitwerte BENCH.PRG*

Tafel 1 Funktioneller Vergleich der Datenbanksysteme

Funktion	dBase II	dBase III	dBase III Plus	Clipper '87	Foxbase	dBase IV	Bemerkungen
Allgemeines							
Kompatibilität	<	B	=>	=>	=>	>>	1)
Konfigurierbarkeit	J	J	J	J	J	J	
Art							
– Interpreter	J	J	J	N	J	J	
– Pseudo-Compiler	N	N	N	N	J	J	
– Maschinencode-Compiler	N	N	N	J	N	N	
Geschwindigkeit							
– maximal	3	1:	1,6	11	17	6	
– typisch	2	1:	1	6	6	2,5	2)
Texteditor integriert?	J	J	J	N	J	J	
Komfort Texteditor	=	B	>	–	>	>>	
maximale Editierlänge in KByte	4	4	4	64	64	3	3)
SQL-Sprache integriert?	N	N	N	N	N	J	
Netzfähigkeit?	N	N	N	J	J	J	
Transaktionskonzept?	–	–	N	N	N	J	4)
Bedarf:RAM > 512 KByte?	N	N	N	N	N	J	5)
Bedarf auf Diskette in MByte	0,13	0,34	0,54	1,2	0,53	2,8	6)
Lauffähig ohne HD?	J	J	J	J	J	N	
Entwicklungssystem							
– Standard, ohne Tools	<	B	=>	<	=>	>>	
– Erweiterung, mit Tools	<	B	=>	=>	=>	>>	
integrierte Generatoren	<	B	=>	<	<	>>	7)
komplette Entwicklungsumgebung?	N	N	N	N	N	J	
Handhabung zur							
– Entwicklung	<	B	=>	<	=>	>>	8)
– Dialoganwendung	<	B	=>	–	=>	=>	9)
Koprozessorunterstützung?	N	N	N	N	J	J	10)
Unterstützung RAM > 640 KByte?	N	N	N	N	J	J	10)
Hilfesystem	<	B	=>	<	=>	>>	
Assistent-Regiezentrum?	N	J	J	N	N	J	
Sicherheit							
– Datenverlust bei Absturz	=	B	=	=	=>	=>	
– gegen unbefugtes Nutzen	=	B	=	=	=	=>	
– im Netzbetrieb	–	–	B	=	=	=>	
Datenbank							
– maximal offene Datenbanken	2	10	10	10	10	10	11)
– offene Dateien, ohne System	15	15	20	15	48	15	11)
– maximale Feldanzahl/Datensatz	32	128	128	1024	128	256	
– maximale Anzahl Feldtypen	3	5	5	5	5	6	
– maximale Datensatzlänge in Byte	1000	4000	4000	4000	4000	4000	
– maximale Datensätze/Datenbank	64 K	1 Mrd.	1 Mrd.	1 Mrd.	1 Mrd.	1 Mrd.	
– Kompatibilität Struktur	<	B	=>	=>	=>	=>	
– Verschlüsselung Inhalt?	N	N	J	N	N	J	
Indexdateien							
einfache Indexdateien	J	J	J	N	J	J	
Mehrfachindexdateien	N	N	N	N	N	J	
Kompatibilität	N	B	=	=	>	=>	12)
Variablen							
maximale Anzahl	64	256	256	2048	3600	2048	
Strukturierte Variablen?	N	N	N	J	J	J	
maximale Strukturdimensionen	–	–	–	1	>2	2	
Bearbeiten Systemvariablen	N	N	N	N	N	J	
Befehle							
maximale Anzahl	<	B	>	>	>	>>	
maximale Anzahl SET-Kommandos	<	B	>	>	>	>>	
Prozedurprogramme?	N	J	J	J	J	J	
maximale Zahl Prozeduren/ Prozedurdatei	–	32	32	bel.	128	1170	
nutzerdefinierte Funktionen?	N	N	N	J	J	J	
Eingabeprüfungen integriert?							13)
– Formate	J	J	J	J	J	J	
– Von-bis-Bereiche	N	N	J	J	J	J	
– konkrete Bedingungen	N	N	N	J	J	J	
Bildschirmsteuerung?							
– Teillöschen Bildschirm	N	J	J	J	J	J	
– BOX-Befehl	N	N	J	J	J	J	
– Pseudowindowing	N	N	N	J	J	J	14)
– Windowing	N	N	N	N	N	J	15)
– Farbfähigkeit des Systems?	N	J	J	J	J	J	
Relationsverknüpfung?							16)
– Linienmodell	N	J	J	J	J	J	
– Baummodell	N	N	N	J	J	J	

```
* Eröffnet wurde im SELECT-Bereich  1 die Datei D1.
* Eröffnet wurde im SELECT-Bereich  2 die Datei D2.
* Eröffnet wurde im SELECT-Bereich  3 die Datei D3.
* (ALIAS  ist jeweils der  volle Dateiname.)
* Relationsbegriff Datei  D1 -->  D2  ist NR2
* Relationsbegriff Datei  D1 -->  D3  ist NR3
SELECT  1

* für dBase IV:
SET  RELATION   TO  NR2  INTO  D2, TO  NR3  INTO  D3

* für Clipper:
SET  RELATION   TO  NR2  INTO  D2,;
             TO  NR3  INTO  D3

* für Foxbase:
SET  RELATION   TO  NR2  INTO  D2
SET  RELATION   TO  NR3  INTO  D3 ADDITIVE
```

Bild 3 Syntax bei Mehrfachrelationen

(die sicher erst mit einem RAM von mehr als 640 KByte entfallen). Das Transaktionskonzept ist besonders wichtig in der Netzarbeit. Nach Programmabstürzen können die Daten auf ihren Ursprungszustand „zurückgerollt" werden, so als wäre die Transaktion nicht durchgeführt worden. Und das mit geringstem Aufwand. Außerdem gibt es bei dBase IV und Foxbase Möglichkeiten, daß erfaßte Daten nicht gepuffert werden brauchen, sondern sofort auf Diskette geschrieben werden können. Die Datenbanken dieser Systeme sind unter Beachtung von Feldtypen, Feldanzahl und Satzanzahl kompatibel; mit Ausnahme von dBase II, dBase II-Datenbanken können aber aufwärts konvertiert werden. Solange nicht im Netz gearbeitet wird oder der Datenzugriff über Datenbanken oder ASCII-Dateien erfolgt, spielt die fehlende Kompatibilität der Indexdateien keine Rolle. Dabei ist zu bemerken, daß Foxbase- und Clipper-Indexdateien effektiver organisiert sind als dBase-Indexdateien. Positiv bemerkbar macht sich dies im geringeren Platzverbrauch und in der höheren Geschwindigkeit.

Verbesserungen in den Systemen Foxbase, Clipper und dBase IV betreffen besonders die Befehle zur

- komplexen Eingangsprüfung bei der Datenerfassung
- Bildschirmsteuerung mit Windowtechnik
- Verknüpfung mehrerer Dateien über abhängige Felder
- Projektierung von menügesteuerten Auswahlbildern
- Einzelblattverarbeitung in Reporten bei dBase IV.

Aufgrund der Geschwindigkeitsvorteile, vor allem von Foxbase und Clipper sind bei der Abarbeitung keine bemerkbaren Wartezeiten nötig. Die Abarbeitung der Programme im RAM, das heißt ohne Nachladen von Harddisk oder Diskette, trägt ebenfalls wesentlich zur Geschwindigkeitssteigerung bei Foxbase, Clipper und dBase IV bei.

Wichtige Arbeitsmittel innerhalb der Datenbanksysteme sind Reporte. Ihre Möglichkeiten, zum Beispiel zum Summieren, wurden kontinuierlich gesteigert und finden bei dBase IV ihren Höhepunkt. Dort sind freie Definition und Gestaltung möglich. Grenzen und offene Wünsche gibt es kaum. Dabei sind auch solche Details berücksichtigt, wie

- Umrandungen von Spalten mit Boxen
- Summierung in Zwischensummen ja und Endsumme nein für ein und denselben Wert, mit beliebig vielen Summenebenen
- Kalkulation über „unsichtbare" Felder.

Urteil insgesamt: Bei der Betrachtung der Funktionalität hat Foxpro eindeutig die Nase vorn, danach folgt dBase IV. Werden die für die Einzelsysteme existierenden Tools mit einbezogen, vermischen sich die Grenzen der Systeme etwas.

Funktion	dBase II	dBase III	dBase III Plus	Clipper '87	Fox-base	dBase IV	Bemer-kungen
interaktive Steuerung	N	N	J	N	J	J	
Menübefehle	N	N	N	J	J	J	17)
dto. verschiedene Menüs	N	N	N	N	N	J	
Lesen/Schreiben ganzer Datensätze (1 Befehl)	N	N	N	N	J	N	
Reporte							
Kompatibilität?	<	B	=>	=>	=>	>	
maximale Anzahl Summenebenen	2	3	3	3	3	bel.	
maximale Anzahl Überschriftszeilen	4	4	4	4	4	>4	
Fußunterschriften?	N	N	N	N	N	J	
Druck							
– spaltenweise	J	J	J	J	J	J	
– zeilenweise	N	N	N	N	N	J	
– Briefform	N	N	N	N	N	J	
– freie Gestaltung?	N	N	N	N	N	J	
Einzelblattverarbeitung?	N	N	N	N	N	J	
Sonstiges							
Abfrage Druckerstatus?	N	N	N	N	J	J	18)
mögliche Druckeranpassung?	N	N	N	N	J	J	
Druckerspooler?	N	N	N	N	J	N	

– kein Vergleich; B Vergleichsbasis; <weniger; = gleich; > besser; => besser, abwärtskompatibel; >> viel besser

1) Kompatibilität > 80 %, aber syntaktische Unterschiede
2) Durchschnittswert für „ganze" Programme
3) Clipper bei Fremdeditor, dBase IV; maximal 32 000 Zeilen, zu maximal 1 KByte
4) automatische Datenkorrektur bis vor dem Zeitpunkt der Fehlersituation, d. h. ohne spezielle Programmierung
5) Komplexe, mit Clipper compilierte Programme benötigen 640 KByte
6) für Entwicklung
7) Report-, Label-, Masken-, View-, Programm-, Abfragegenerator
8) Berücksichtigung auch von Interpreter/Compiler
9) Komfort der Anwenderprogramme abhängig von kronkretem Programm
10) automatisch Ja/Nein
11) Gesamtanzahl offener Dateien bis MS-DOS 3.3 ≦20 (betriebssystemabhängig), bei dBase III Plus mit Zusatzmodul ≦ 25 Dateien, bei Foxbase standardmäßig ≦ 48 Dateien
12) mit Zusatzmodul erreichbar bei Clipper und Foxbase; automatische Anpassung, daher abwärtskompatibel
13) Eingabefeld kann bei Nichterfüllung nicht verlassen werden
14) Einlesen/Abspeichern ganzer Bildschirminhalte, Überlagern möglich (programmierbar wie „echtes" Windowing, aber der Aufwand muß beachtet werden)
15) echtes Öffnen/Schließen/Einlesen/Abspeichern von Fenstern, automatisches Retten des Bildschirminhaltes
16) Linienmodell: Anknüpfen 1 Kindsdatei an Vaterdatei usw., Baummodell: Anknüpfen mehrerer Kindsdateien gleichzeitig an die Vaterdateien
17) ohne komplizierte Programmierung
18) Drucker angeschlossen und empfangsbereit? (Prüfung vor dem Druck, aber nur bei paralleler Schnittstelle möglich)

Tafel 2 Laufzeitvergleich (für ausgewählte Funktionen)

System / Funktion	dBase II V.2.41	dBase III V.1.10	dBase III Plus V.1.0	Clipper '87 V.2.24	Foxbase V.2.1	dBase IV V.1.0
Zählen von 1 bis 10 000	126	177	221	16	24	34
1000 Datensätze anfügen	60	174	211	16	10	29
Indizieren Datei mit 1000 Sätzen	60	40	24	17	12	22
Suchen in Datei, 1000mal hintereinander	81	57	52	13	15	19
Ersetzen Textfeld, 1000mal	55	135	162	36	22	52

Laufzeitwerte in Sekunden maschinell ermittelt mit der Testdatei nach Tafel 1 und Programm nach Bild 1
Ausnahme bei dBase II: Programm leicht modifiziert und Zeiten manuell ermittelt
Schwankungsbreite der Werte bis > 20 % (Ursachen siehe Text)

Laufzeitvergleich

Laufzeitmängel spielen nicht nur in der subjektiven Akzeptanz eine Rolle. Laufzeitgrenzbedingungen existieren auch objektiv. Nach meinen Erfahrungen sollten beispielsweise Grenzzeiten von 0,1 bis 0,5 Sekunden in folgenden Funktionen eingehalten werden:
– Bildschirmaufbau
– Suchen bestimmter Datensätze in einer Datei (unabhängig von der Gesamtanzahl der Datensätze!)
Hier liegen bei dBase eindeutige Schwächen, und die genannten Forderungen sind bis dBase III Plus kaum einzuhalten. Hier sind auch die Ursachen für die Entwicklung von Foxbase und Clipper zu suchen. Beide arbeiten, obwohl sie Interpreter/Pseudocompiler und echter Compiler sind, etwa gleich schnell wie andere Maschinensprachenprogramme (siehe Tafel 2). Nach Herstellungsaussage ist Foxpro noch dreimal schneller als Foxbase. Die absoluten Werte des Laufzeitvergleiches aus Tafel 3 sind aber nicht aussagefähig, da Abhängigkeiten existieren, die nichts mit der Leistungsfähigkeit der Software zu tun haben. Im einzelnen wären das zum Beispiel:

Hardwareabhängige Faktoren
– Prozessortyp
– Verarbeitungsbreite (8, 16 oder 32 Bit)
– Vorhandensein eines Koprozessors und Unterstützung der Koprozessorarbeit durch die Software
– RAM 512 oder 640 KByte (Foxbase ist mit 640 KByte etwa 1/3 schneller als mit 512 KByte)
– Zugriffsgeschwindigkeit von Harddisk und Floppy
– Fabrikatbedingte Unterschiede (z. B. Positionierzeiten bei Harddisks)

Softwareabhängige und organisatorische Faktoren
– Abarbeitung von Programmen zeilenweise oder programmweise
– dynamische Ausnutzung des verfügbaren Speichers (RAM)
– Pufferung der Daten und Pufferverwaltung
– Effektivität der Laufzeitorganisation durch verwendete Basisroutinen
– Effektivität der Arbeit mit Indexdateien
– subjektive Programmierleistung in einzelnen Programmzweigen
– Organisation der Festplatte, das heißt, wie sind die benutzten Dateien auf der HD verteilt? Im ganzen Stück oder wild verteilt? Letzteres führt zu Geschwindigkeitsverlust von bis zu 50 %.
Zur Steigerung der Geschwindigkeit allgemein kann folgendes getan werden: Mindestens einmal alle 4 Wochen sollte die Harddisk reorganisiert werden (Laufzeit für eine Reorganisation: ca 1 h). Sequentiell suchende Befehle (Locate, Count usw.) sollten nach Möglichkeit nicht verwendet oder durch indexsequentielle ersetzt werden, und Dateien können zwecks Einsparung von Berechnungszeit, temporären Dateien und Datenredundanz über Relationen verbunden werden (dadurch sind z. B. Reporte über 2, 3, 4 und mehr Dateien ohne Programmierung möglich); also ein Vorteil für Foxbase, Clipper und dBase IV, da hier Dateien „baumartig" verbunden werden können ... Der Einsatz eines Koprozessors bringt nur bei Foxbase (Funktion: Suchen) und dBase IV (Funktionen: Indizieren und Suchen) Laufzeitvorteile (bis zu 30 %).
Beim Laufzeittest haben Foxbase und Clipper die Nase vorn, gefolgt von dBase IV. Leider stand zum Testzeitpunkt noch kein Compiler für dBase IV zur Verfügung.
Geschwindigkeitsvorteile traditioneller Programmiersprachen gegenüber Foxbase und Clipper bestehen kaum (da die in diesen Systemen verwendeten Routinen effektiv sind und in C oder Assembler geschrieben worden sind).

Versuch einer Empfehlung

Der Vergleich von Funktionen und Laufzeiten liefert keine eindeutigen Ergebnisse, da wichtige Anwendungsbedingungen fehlen, das heißt, die konkrete Hardware und die Betriebssystemsoftware ist mit einzubeziehen. Daher sollte eine Empfehlung wie folgt aussehen: Insgesamt von Vorteil ist die Nutzung von dBase II für 8-Bit-PCs und von Foxpro für 16-Bit-PCs.
Die vorgestellten Datenbanksysteme bieten entwicklungsfreundliche Eigenschaften sowie für kleine Probleme Möglichkeiten im Dialog. Jedoch auch komplexe Probleme, wie die betriebliche Materialwirtschaft, sind beherrschbar. Dabei ist aber zu beachten, daß mit Clipper compilierte Programme in der Größe unter 320 KByte bleiben sollten (als EXE-Datei), wenn die Größe des Hauptspeichers nur 512 KByte beträgt. Größere Programme erfordern mehr Hauptspeicher zum Arbeiten. Dieses Problem tritt bei Foxbase und bis dBase III Plus nicht auf. Allerdings sollten Hardwareprobleme auch in der DDR der Vergangenheit angehören. Ein Hauptspeicher von mehr als 640 KByte und eine Harddisk sollten für 16-Bit-PCs zum Standard gehören. Die Entwicklung geht unaufhaltsam weiter. Nach Sprachen der 4. Generation, zu denen dBase und Foxbase gehören, kommen unweigerlich die der 5. Generation.

Literatur

/1/ Codd, E. F.: A Relation Model for Large Chared Data Banks. Comm. ACM, Vol. 13, 6/1970

/2/ Eggerichs, W.: dBase 2: Einführung und Programmierung. Berlin: Verlag Technik 1986

/3/ Grafik, W.: dBASE III. Berlin. Verlag Technik 1989

/4/ Grafik, W.: dBASE III im Vergleich. Mikroprozessortechnik, Berlin 2 (1988) 10, S. 294

/5/ Grafik, W.: Erfahrungen beim Übergang von dBase II zu dBase III. Mikroprozessortechnik, Berlin 2 (1989) 3, S. 72

/6/ Grafik, W.: Fortschritte in dBase III Plus. Mikroprozessortechnik, Berlin 3 (1989) 4, S. 118

/7/ CLIPPER – Compiler Anwender Handbuch. Culver City: Nantucket Corporation 1985

/8/ Dahlmann, P.: Zweimal mehr als das Vorbild. Chip. München 11 (1988) 10, S. 264

/9/ Herbert, H.-P.: ADAD9, DATAFLEX, FOXBASE – Datenbanken für Programmentwickler. mini Micro magazin, Heidelberg 4 (1988) 3, S. 80

/10/ Herbert, H.-P.: Datenbanksystem FOXBASE Plus 2.00: Auf Geschwindigkeit getrimmt. mini Micro magazin, Heidelberg 4 (1988) 11

/11/ Herbert, H.-P.: Perfekte Datenbank. Chip, München 11 (1989) 1, S. 155

/12/ Weber, P.: Redabas- Laufzeiten unter Betriebssystemen für BC A 51xx und PC 1715. rechentechnik/datenverarbeitung, Berlin 24 (1987) 4, S. 31

/13/ Fischer, M.: dBase IV. Mikroprozessortechnik, Berlin 3 (1989) 2, S. 44

/14/ Der DATA-Becker Führer zu dBASE IV. Düsseldorf: DATA-Becker Verlag 1989

/15/ Kaier, E.: dBASE IV griffbereit. Braunschweig: Vieweg & Sohn Verlagsgesellschaft mbH 1989

/16/ Pfeiffer, St.: Foxbase Plus 2.1 – Zwischen dBASE III Plus und dBASE IV. Dos Test, Haar 1 (1989) 9, S. 104

/17/ Das Vorbild und die Nachfolger. PC Praxis, Düsseldorf (1990) 1, S. 34

/18/ Wettstreit der Datenjongleure. Dos Test, Haar 2 (1990) 1, S. 90

/19/ Wer ist der Schnellste im ganzen Land? Dos Test, Haar 2 (1990) 2, S. 114

Gründungsfieber — Teil 2

Der Traum von der Selbständigkeit

Dieter Butscher

Eine OHG, GmbH, KG, GmbH & Co KG, GmbH & Still zu gründen – soweit möchte mancher Neu-Unternehmer anfangs gar nicht gehen. Doch bereits die lockere unternehmerische Zweierbeziehung bleibt nicht ohne Folgen. Deren Kind hört auf den Namen Gesellschaft bürgerlichen Rechts und kommt schneller in die Welt als erwartet.
Bei der OHG, der KG sowie der weitverbreiteten GmbH kommen im Gegensatz zur Gründung einer Gesellschaft bürgerlichen Rechts, die formfrei und sogar durch bloße mündliche Absprache möglich ist, Zwänge hinzu, die sich zum bürokratischen Hürdenlauf ausweiten können.

Bürgerliche Gesellschaft

Wer sein Unternehmen nicht als Einzelgänger, sondern zusammen mit anderen aufbaut, kommt nicht umhin, eine Gesellschaft zu gründen. Dabei wird in der Praxis oft blind losgelegt. Über die richtige Gesellschaftsform oder gar den Abschluß eines schriftlichen Gesellschaftsvertrages macht man sich wenig Gedanken. Erst wenn größere Auseinandersetzungen zwischen den Geschäftspartnern anstehen, kommt die Frage nach dem Gesellschaftsvertrag auf.
Üben zwei oder mehrere Freiberufler oder Kleingewerbetreibende gemeinsam ein Unternehmen aus, besteht eine Gesellschaft bürgerlichen Rechts (GbR). Der Gesellschaftsvertrag kann auch mündlich geschlossen werden oder „konkludent" durch das gemeinsame Betreiben eines Unternehmens zustande kommen. Damit wird bereits deutlich, daß die Gründung einer GbR denkbar einfach ist.
Dazu genügt ein Zusammenschluß mehrerer Personen zu einem gemeinsamen Zweck. Dieser liegt bei Gründung eines Unternehmens immer vor. Eine Anmeldung zum Handelsregister ist ebensowenig erforderlich wie ein schriftlicher Gesellschaftsvertrag oder eine bestimmte Stammkapitalsumme. Es bleibt bei der Gewerbeanmeldung und der Anzeige an das Finanzamt.
Man sollte sich in der Gründungsphase dennoch die Zeit zum Abschluß eines Gesellschaftsvertrages nehmen. Wer nämlich bei einem späteren Zerwürfnis Ansprüche an die anderen Gesellschafter stellt und sich dabei auf getroffene Vereinbarungen beruft, muß diese beweisen. Mündliche Absprachen sind Quelle lang anhaltender Probleme und Streitigkeiten.

Auch wenn kein mündlicher oder schriftlicher Gesellschaftsvertrag geschlossen worden ist, besteht die GbR dennoch. Es gelten dann die gesetzlichen Vorschriften (§§ 705 ff. BGB). Diese stellen nur einen pauschalen Rahmen zur Verfügung und führen bei einer Auseinandersetzung in nicht wenigen Fällen zu Ergebnissen, die niemand so wollte.

Zankäpfel

Dabei mag man am Anfang nur ungern an mögliche Streitigkeiten denken und weist den Gedanken an ein Zerwürfnis weit von sich. Daß dies nicht der Erfahrung entspricht, muß nicht gesondert erwähnt werden. Daher muß ein Gesellschaftsvertrag Regelungen darüber enthalten, welche Einlagen die Gesellschafter zu erbringen haben, welche Gegenstände der einzelnen Gesellschafter in das Vermögen der Gesellschaft übergehen oder Sondervermögen bleiben, wer die Gesellschaft vertreten kann, wer die Führung übernimmt, ob die Gesellschaft bei Ausscheiden eines Gesellschafters fortgesetzt wird, wie sich der Abfindungsanspruch des Ausscheidenden berechnet.
Einer der wichtigsten Punkte, der unbedingt einer Regelung bedarf, ist die Vertretungsbefugnis. Sie bestimmt, wer für die Gesellschaft Verträge abschließen darf und damit alle (!) Gesellschafter nicht nur berechtigt, sondern auch verpflichtet. Wird einem einzelnen Gesellschafter die Einzelvertretungsbefugnis eingeräumt, kann er nicht nur Verträge für die Gesellschaft abschließen, sondern auch Verpflichtungen (etwa die Aufnahme eines Kredits) für alle Gesellschafter als Gesamtschuldner begründen.
Wenn alle Gesellschafter nur gemeinschaftlich Verträge abschließen dürfen („gemeinschaftliche Vertretungsbefugnis"), treten ebenfalls Probleme auf. Erkrankt ein einziger, liegt der Betrieb lahm. In der Regel hilft das Eingrenzen der Vertretungsbefugnis auf bestimmte Geschäftsarten, zum Beispiel Alltagseinkäufe und -geschäfte, auf eine Vertretung nur jeweils zu zweit oder zu dritt oder auf eine gemeinschaftliche Vertretung aller Gesellschafter bei gewichtigen Verpflichtungen.
Für Verbindlichkeiten der Gesellschaft haften alle Gesellschafter als Gesamtschuldner nicht nur mit dem Vermögen der Gesellschaft, sondern auch mit ihrem jeweiligen Privatvermögen. Der Gläubiger kann sich einen einzelnen herauspicken, gegen ihn die Forderung in voller Höhe durchsetzen und, wie die Juristen formulieren, „in sein Privatvermögen vollstrecken". Denn die GbR stellt rechtlich gesehen keine eigenständige Rechtsperson dar, sondern nur einen Zusammenschluß einzelner Personen.

Begrenzung

Daher trifft die vertragliche Verpflichtung nicht die ohnehin nicht rechtsfähige Gesellschaft, sondern die einzelnen Gesellschafter persönlich. Aus einem abgeschlossenen Kreditvertrag haftet jeder einzelne Gesellschafter in voller Höhe mit seinem gesamten Privatvermögen – dies selbst dann noch, wenn er bereits aus der Gesellschaft ausgeschieden ist oder die Gesellschaft sich aufgelöst hat. Jeder ist auf Grund seiner persönlichen gesamtschuldnerischen Haftung für die Schulden der GbR auf Gedeih und Verderb mit der Gesellschaft verbunden!
Eine Haftungsbegrenzung gegenüber den Gläubigern auf das Vermögen der Gesellschaft ist durch Vereinbarung im Gesellschaftsvertrag möglich, muß jedoch den Gläubigern vor Abschluß des Vertrages zur Kenntnis gegeben werden oder auf sonstige Weise bekannt sein. Dadurch läßt sich zumindest bei Abschluß größerer Geschäfte das Schlimmste vermeiden.
In steuerlicher Hinsicht unterscheidet sich die GbR wenig von dem Einzelunternehmen. Handelt es sich bei dem Betrieb um ein Gewerbe, ist Gewerbesteuer zu entrichten. Die Gesellschaft als solche unterliegt nicht der Einkommenssteuerpflicht, dagegen die Gewinnanteile der einzelnen Gesellschafter. Hierzu wird am Ende eines jeden Geschäftsjahres der von der Gesellschaft erzielte Gewinn oder Verlust in Form einer einheitlichen Gewinnfestsetzung ermittelt. Je nach vereinbartem Verteilungsschlüssel entfallen auf jeden einzelnen Gesellschafter Gewinn- oder Verlustanteile, die von diesem als „Einkünfte aus Gewerbebetrieb" beziehungsweise „Einkünfte aus selbständiger Tätigkeit" der Einkommenssteuer zu unterwerfen sind.
Immer wieder hört man davon, daß Existenzgründer Hals über Kopf versuchen, in Zukunft unabhängig ihr täglich Brot zu erwerben. Im Eifer des Gefechts denken sie dabei nicht an naheliegende Probleme: Angestellte sind kranken-, renten- und arbeitslosenversichert. Die Hälfte der Beiträge übernimmt der Arbeitgeber. Selbständige haben die Kosten in voller Höhe selbst zu tragen. Wer eine Familie hat, kommt nicht umhin, auch deren soziale Sicherheit zu gewährleisten.
Nach einer Kündigung gewähren die gesetzlichen Krankenkassen zwar noch eine „Nachversicherungsfrist" von vier Wochen, danach steht der optimistische Gründer in seinem Fieber ohne Absicherung allein auf weiter Flur. Einen Mindestbeitrag von gut 300 DM berappen Sie monatlich der Krankenkasse (ohne Anspruch auf Krankengeld), auch

Mustervertrag einer Gesellschaft bürgerlichen Rechts

§ **1** Name, Sitz
Die Vertragschließenden errichten eine Gesellschaft bürgerlichen Rechts.
Die Gesellschaft für die Bezeichnung „Müller, Droste und Henning DV-Vertrieb".
Sitz der Gesellschaft ist Hannover.

§ **2** Gegenstand des Unternehmens
Gegenstand des Unternehmens ist der Handel mit Hard- und Software.

§ **3** Einlage
Die Gesellschafter Dieter Müller, Frank Droste und Herbert Henning erbringen eine Einlage von je 20000 DM.
Die Einlage ist von jedem Gesellschafter auf das zu eröffnende Konto der Gesellschaft bis spätestens zum 1. Juli 1990 einzuzahlen.

§ **4** Dauer der Gesellschaft
Die Gesellschaft beginnt am 1. Juni 1990. Die Dauer der Gesellschaft ist unbestimmt; die Gesellschaft kann mit einer Frist von sechs Monaten zum Schluß eines jeden Geschäftsjahres gekündigt werden. Das Geschäftsjahr ist das Kalenderjahr.

§ **5** Geschäftsführung
Zur Geschäftsführung und Vertretung sind die Gesellschafter nur gemeinschaftlich berechtigt und verpflichtet.
Im Rahmen des alltäglichen Geschäftsbetriebs ist jeder Gesellschafter zum Abschluß von Verträgen bis zu einem Wert von 5000 DM einzelvertretungsberechtigt.
Geschäftsführung und Vertretungsbefugnis können einem Gesellschafter durch Gesellschafterbeschluß entzogen werden, wenn ein wichtiger Grund vorliegt.

§ **6** Informations- und Kontrollrecht
Jeder Gesellschafter, auch wenn er von der Geschäftsführung ausgeschlossen ist, kann sich in allen Angelegenheiten der Gesellschaft persönlich unterrichten, die Geschäftsbücher und Unterlagen der Gesellschaft einsehen.

§ **7** Gesellschafterversammlung, Beschlüsse
Mindestens einmal im Geschäftsjahr findet eine Gesellschafterversammlung statt. Beschlüsse werden mit einfacher Mehrheit aller Gesellschafter gefaßt. Zu einer Änderung des Gesellschaftsvertrages und der Aufnahme neuer Gesellschafter bedarf es der Zustimmung aller Gesellschafter.

§ **8** Jahresabschluß
Eine Bilanz ist nebst Gewinn- und Verlustrechnung von den geschäftsführenden Gesellschaftern jährlich aufzustellen und von allen Gesellschaftern zu genehmigen.

§ **9** Gewinn- und Verlustverteilung, Entnahmen
Die Gesellschafter sind zu gleichen Teilen am Gewinn und Verlust beteiligt.
Die Höhe der Entnahmen wird durch Beschluß der Gesellschafterversammlung festgelegt.

§ **10** Ausscheiden eines Gesellschafters
Bei Kündigung sowie Ausschluß eines Gesellschafters wird die Gesellschaft nicht aufgelöst, sondern unter den übrigen Gesellschaftern fortgesetzt.
Die Kündigung eines Gesellschafters ist unter Einhaltung einer Frist von sechs Monaten zum Ende eines Geschäftsjahres zulässig. Unberührt hiervon bleibt das Recht eines jeden Gesellschafters zur Kündigung aus wichtigem Grund.
Der Ausschluß eines Gesellschafters ist nur bei vorsätzlicher, grober Pflichtverletzung zulässig.

§ **11** Aufnahme neuer Gesellschafter
Die Aufnahme neuer Gesellschafter bedarf der Zustimmung aller bisherigen Gesellschafter. In dem Aufnahmevertrag sind die Höhe der von dem neuen Gesellschafter zu erbringenden Einlage, seine Gewinn- und Verlustbeteiligung sowie die Geschäftsführungs- und Vertretungsbefugnisse zu regeln.

§ **12** Auseinandersetzung
Im Auseinandersetzungsfall (bei Ausscheiden eines Gesellschafters aufgrund Kündigung oder Ausschluß) ist unverzüglich eine Abfindungsbilanz zu erstellen. In dieser Bilanz sind alle Vermögensgegenstände mit ihrem Zeitwert anzusetzen.
Das Auseinandersetzungsguthaben ist in drei gleichen Jahresraten auszuzahlen.

§ **13** Schlußbestimmung
Alle Änderungen dieses Gesellschaftsvertrages sind nur wirksam, wenn sie schriftlich erfolgen.
Hannover, den 5. Juni 1990

Ein Vertrag sichert selbst einen losen Zusammenschluß mehrerer Personen zuverlässiger ab, als dies mündliche Absprachen oder allgemeine gesetzliche Bestimmungen vermögen.

dann, wenn sie *kein* Einkommen haben! Sehen sie einmal auf Ihren Lohnzettel, was eine Weiterversicherung im Gesundheits- und Rentenbereich kosten wird!

Offener Handel

Die Gesellschaftsform der Offenen Handelsgesellschaft (OHG) steht bei möglichen Kreditgebern, und hier vor allem bei den Banken, in hohem Ansehen. Kein Wunder – ähnlich wie bei der Gesellschaft bürgerlichen Rechts haften nämlich alle Gesellschafter nicht nur mit dem Gesellschaftsvermögen, sondern in vollem Umfang auch mit ihrem Privatvermögen. Nicht zuletzt dieser Haftungsrisiken wegen trifft man die Rechtsform der OHG in der Computerbranche nicht allzu häufig an.
Die Gründung einer OHG setzt ein „vollkaufmännisches Unternehmen" voraus. Freiberuflern und Kleingewerbetreibenden (sogenannten Minderkaufleuten) bleibt sie versagt. Damit scheidet die OHG in den meisten Fällen auch als Einstiegsform für Existenzgründer aus. Nur wer sein Geschäft gleich im großen Stil starten will und in der Lage ist, einen vollkaufmännischen Betrieb einzurichten, mag die Gründung einer OHG in Betracht ziehen.
Voraussetzung für das Entstehen einer OHG ist die Eintragung im Handelsregister. Die Gesellschaft muß eine Firma (Name der OHG) führen und unterliegt denselben handelsrechtlichen Vorschriften wie Vollkaufleute. Ein bestimmtes Stammkapital ist zur Gründung nicht erforderlich. Die Eintragung in das Handelsregister schafft eine vom Gesetzgeber gewollte Publizität und Offenkundigkeit. Jedermann kann sich per Einblick in das Handelsregister Klarheit darüber verschaffen, welche Gesellschafter hinter der OHG stehen.
Wer ein unsicheres und risikoreiches Unternehmen aufbauen will, sollte sich die Gründung schnell aus dem Kopf schlagen. Gegen sie spricht nicht nur die volle persönliche Haftung aller Gesellschafter. Die OHG belastet den ausscheidenden Gesellschafter auch mit unangenehmen Nachwehen. Wer abspringen möchte, weil die Überschuldung oder der Konkurs bereits in Sicht ist, kann sich damit noch nicht in Sicherheit wähnen. Er haftet weiter persönlich für die Schulden der OHG, die zur Zeit seines Ausscheidens bestehen – und zwar fünf Jahre lang, gerechnet ab Eintragung seines Ausscheidens im Handelsregister.

Bäumchen, wechsel dich

Wer umgekehrt in eine bestehende OHG als neuer Gesellschafter eintreten will, sollte sich über die Vertrauenswürdigkeit seiner Kollegen und die Liquidität der Gesellschaft vergewissern. Der Novize haftet nämlich für die bereits bestehenden Verbindlichkeiten der OHG in vollem Umfang mit seinem gesamten Vermögen. Häufiger Partnerwechsel ist aus diesem Grund bei der OHG nicht angesagt. Nur wer einen oder mehrere feste und vertrauenswürdige Partner hat, kann sich überlegen, ob er sein Handelsgewerbe langfristig in der Form der OHG betreiben will.
Trotz dieser offenkundigen Nachteile: In manchen Fällen ist die Gründung einer OHG unumgänglich. So kann eine Gesellschaft bürgerlichen Rechts quasi zu einer OHG heranwachsen. Dies ist der Fall, wenn sich das zunächst als Kleingewerbe betriebene Unternehmen zum vollkaufmännischen Betrieb mausert. Mit Erreichen einer bestimmten Umsatzzone wird aus der GbR, die ein Handelsgewerbe betreibt, schließlich eine OHG. Für die Gesellschaft führt dies zu der Verpflichtung, die bereits faktisch bestehende OHG zur Eintragung im Handelsregister anzumelden. Es gelten hier dieselben Grundsätze wie beim Übergang vom einzelkaufmännischen Kleingewerbe zum Vollkaufmann.

Steuern

In steuerlicher Hinsicht läßt sich folgendes anmerken: Die OHG ist umsatz- und gewerbesteuerpflichtig. Sie unterliegt aber nicht

der Vermögens- und Einkommenssteuer. Das Vermögen und die von der OHG erzielten Gewinne beziehungsweise Verluste werden wie bei der Gesellschaft bürgerlichen Rechts den einzelnen Gesellschaftern zugerechnet. Diese werden dann nach einheitlicher und gesonderter Gewinnfeststellung vom Finanzamt zur Einkommenssteuer und Vermögenssteuer herangezogen.
Wieviel Einkommenssteuer jeder Gesellschafter zu entrichten hat, folgt aus seinen individuellen Verhältnissen. Lästig fällt vor allem die Gewerbesteuer ins Gewicht. Hier werden Zahlungen an die Gesellschafter wie Tätigkeitsvergütungen nicht als gewerbesteuermindernd angesehen. Ein Gewerbesteuerfreibetrag wird nur auf die ersten 36000 DM des Gewerbeertrags gewährt. Dieser ist schnell überschritten.

Treue Eheleute

Die Kommanditgesellschaft (KG) zeigt eine Variation der OHG. Während bei der OHG alle Gesellschafter persönlich haften, übernimmt diese Rolle bei der KG der Komplementär („persönlich haftender Gesellschafter"). Die anderen Gesellschafter heißen Kommanditisten. Sie haften nur bis zu der Höhe der von ihnen erbrachten Einlage. An das Privatvermögen der Kommanditisten kommen die Gläubiger nicht heran.
Die Gründung einer KG bietet sich vor allem dann an, wenn Kapitalgeber vorhanden sind, die zu einer Beteiligung an dem Unternehmen mit Gewinn und Verlust bereit sind, aber kein persönliches Haftungsrisiko tragen möchten. Als Ausgleich für das übernommene persönliche Haftungsrisiko bleibt der Komplementär Steuermann an Bord. Entscheidungsgewalt und Geschäftsführung liegt nur bei dem oder den persönlich haftenden Gesellschaftern. Steuerlich behandelt das Finanzamt die KG genauso wie die OHG.
OHG-Gesellschafter und KG-Komplementäre erkennt man im Privatleben häufig daran, daß sie kaum Eigentum haben und ein scheinbar intaktes Eheleben führen. Wer die vorangegangenen Zeilen gelesen hat, wird unschwer einsehen, warum dieser Personenkreis beizeiten bedeutende Besitztümer dem Ehepartner überschrieben hat. Wie viele Ehen könnten gerettet werden, wenn der Herrgott allen Menschen die Verpflichtung auferlegt hätte, den Rotstift so zu führen, wie dies Komplementäre tun müssen.

GmbH

Weit verbreitete und beliebteste Form: die Gesellschaft mit beschränkter Haftung (GmbH). Die entscheidenden Vorteile zeigen sich in der Begrenzung des Haftungsrisikos und der rechtlichen Selbständigkeit. Als rechtliche „Persönlichkeit" führt sie ein eigenständiges Leben. Die Gesellschafter können wechseln, ohne den Bestand der GmbH zu berühren. Jeder Gesellschafter hat bei Gründung beziehungsweise Aufnahme in die GmbH eine Einlage zu erbringen. Für Verbindlichkeiten haften nicht die Gesellschafter persönlich, sondern die GmbH – begrenzt auf das Gesellschaftsvermögen.
Damit haftet jeder Gesellschafter nur bis zu der Höhe der von ihm erbrachten Einlage. Geht das Unternehmen baden und muß die

---

Förderungsmöglichkeiten

Existenzgründungsberatung
60 Prozent der Beratungskosten bis zu einem Höchstbetrag von 2500 DM werden beim Aufbau einer selbständigen Existenz durch Neugründung von Unternehmen, Betriebsübernahme oder tätige Beteiligung an einem bestehenden Unternehmen erstattet. Die Beratung vor Aufbau der selbständigen Existenz soll Klärung über die wirtschaftliche, finanzielle und technische Tragfähigkeit des Projekts verschaffen. Industrie- und Handelskammern, Handwerkskammern und Unternehmerverbände bieten oft kostenlose Beratungen. Die Förderungsmittel werden vom Bundesministerium für Wirtschaft vergeben.

Existenzaufbauberatung
Ebenfalls 60 Prozent der Beratungskosten (höchstens 3000 DM) werden für Beratungen in der Aufbauphase des Unternehmens nach Existenzgründung erstattet, sofern den Förderungsrichtlinien entsprochen wird. Die Mittel kommen ebenfalls vom Bundesministerium für Wirtschaft und sollen dazu beitragen, die wirtschaftlichen, technischen und organisatorischen Schwierigkeiten in der Anlaufphase des Unternehmens zu überwinden.

Ansparförderung
Der Bund gewährt einen Ansparzuschuß von 20 Prozent der Sparleistungen (maximal 10000 DM) zum Zwecke der Existenzgründung im gewerblichen Bereich (produzierendes Gewerbe, Handwerk, Handel und Kleingewerbe, nicht für Freiberufler), wenn folgende Voraussetzungen vorliegen:
– Die Sparzeit muß mindestens drei und darf höchstens zehn Jahre betragen.
– Die Sparleistungen müssen als Eigenkapital in das neugegründete Unternehmen eingebracht werden.
Der Ansparzuschuß kann erst beantragt werden, wenn die Existenzgründung erfolgt ist und die Geschäfte aufgenommen werden.

Eigenkapitalhilfe
Existenzgründer, welche das notwendige Eigenkapital nicht aufbringen, können mit Hilfe dieses Programms die eigenen Mittel, die mindestens 12 Prozent der Investitionssumme betragen sollen, um bis zu 40 Prozent des errechneten Kapitalbedarfs aufstokken. Förderungsfähige Projekte sind Existenzgründungen im gewerblichen und freiberuflichen Bereich, Betriebsübernahmen oder tätige Beteiligungen.
Voraussetzung ist, daß ohne die Eigenkapitalhilfe eine erfolgversprechende Existenzgründung wesentlich erschwert würde, das Vorhaben eine tragfähige wirtschaftliche Existenz erwarten läßt, die Investitionssumme mindestens 40000 DM beträgt und mit dem Vorhaben zum Zeitpunkt der Antragstellung noch nicht begonnen worden ist.
Gewährt werden maximal 30000 DM bei einer Laufzeit von 20 Jahren. Davon bleiben zehn Jahre tilgungs- und zwei Jahre zinsfrei.

ERP
Dieses Programm entspringt noch den Mitteln des Marshallplans (ERP: European Recovery Program). Gefördert werden aus dem Sondervermögen des ERP Nachwuchskräfte (Alter: 21–50 Jahre) durch Gewährung zinsgünstiger Darlehen.
Folgende Vorhaben im gewerblichen Bereich sind förderungsfähig: Investitionen zur Errichtung und zum Erwerb von Betrieben (auch Folgeinvestitionen innerhalb von drei Jahren nach Aufnahme der Geschäftstätigkeit), tätige Beteiligungen oder Beschaffung eines ersten Warenlagers beziehungsweise einer Büroausstattung.
Darlehen werden bis zu einem Höchstbetrag von 30000 DM gewährt, bei einer Laufzeit zwischen 10 und 15 Jahren. Sie bleiben zwei Jahre tilgungsfrei. Der Zinssatz von 5,5 Prozent ermäßigt sich für Berlin auf 3,5 Prozent.
Neben der Gewährung von Existenzgründungsdarlehen gibt es noch die Möglichkeit der Bürgschaftsübernahme zur Beschaffung für Kredite – zum Beispiel beim Aufbau einer freiberuflichen Existenz.

Programm für technologieorientierte Unternehmensgründungen
Hierfür gibt es vom Bundesministerium für Wirtschaft und den Landesregierungen Förderungsprogramme im Bereich technologisch-innovativer Unternehmensgründungen. Beratungen, finanzielle Förderungen von Forschung und Entwicklung und Beschaffung von Risikokapital sind Bestandteile dieser Programme.
Anlaufstelle zur Erkundung der diversen Förderungsmöglichkeiten sind Banken, Wirtschaftsministerien des Bundes und der Länder, Industrie- und Handelskammern, Innungen und Verbände. Volkshochschulen bieten Existenzgründungs-Workshops an, in einigen Städten (zum Beispiel Berlin) finden jährlich Gründertage statt. Hier kann man sich durch Teilnahme an Seminaren und Vorträgen einen recht guten Überblick über Förderungsmöglichkeiten verschaffen.

Das Bundesministerium für Wirtschaft
– Referat Öffentlichkeitsarbeit –
Villemombler Straße 76
5300 Bonn
versendet kostenlose Informationsbroschüren zu folgenden Themen:
– Förderungsmaßnahmen des Bundes für mittelständische Unternehmen, freie Berufe und Existenzgründer
– Ansparförderung zur Existenzgründung in der gewerblichen Wirtschaft
– Das ERP.

Gesellschaft Konkurs anmelden, verflüchtigt sich im schlimmsten Fall die Einlage. Das private Fahrzeug des Gesellschafters ist dem Zugriff der Gläubiger entzogen.
Im Bereich riskanter und gewagter Geschäfte ist daher die GmbH die bevorzugte Organisationsform. Durch Gründung einer GmbH lassen sich nicht nur vertraglich eingegangene Verpflichtungen auf das Gesellschaftsvermögen begrenzen, auch weitere Haftungstatbestände (wie die Produkthaftung) sind auf das Gesellschaftsvermögen eingegrenzt.

Kapital

Da mit der Gesellschaftsform der GmbH in der Praxis reichlich Schindluder getrieben wird, lassen sich Kreditgeber und Versicherungen meist durch private Schuldübernahmen, private Bürgschaften der Gesellschafter oder Bestellung zusätzlicher Sicherheiten absichern.
Für chronisch finanzschwache Mitbürger wird die Gründung einer GmbH bereits daran scheitern, daß hierfür ein Stammkapital von mindestens 50 000 DM aufgebracht werden muß.[1] Vor Eintragung in das Handelsregister muß von diesem Betrag mindestens die Hälfte eingezahlt werden. Den Nachweis dieser Einzahlung verlangt das Registergericht.
Statt als „Bargründung" kann eine GmbH auch im Wege der „Sachgründung" erfolgen. Diese kommt in Betracht, wenn erhebliche Sachwerte (Grundstücke, Betriebsgegenstände, Werkzeuge, Maschinen, Fahrzeuge) in die Gesellschaft eingebracht werden können, deren Wert insgesamt 50 000 DM erreicht. Dann verlangt das Registergericht einen Sachgründungsbericht, aus dem sich die angemessene Bewertung der erbrachten Sacheinlagen ergibt.
Der Blick des Registergerichts auf diesen Sachgründungsbericht ist kritisch. Wer schummeln will, indem er einzelne Gegenstände überbewertet, wird schnell enttarnt. Eine Sachgründung kann jedoch den Übergang einer Gesellschaft bürgerlichen Rechts zur GmbH erleichtern. Statt einer Neugründung durch Erbringung der Bareinlagen läßt sich so das bisherige Betriebsvermögen der GbR in die GmbH einbringen.
Die Gründung einer GmbH ist vertrackt: Der Gesellschaftsvertrag muß schriftlich ausgearbeitet und unterzeichnet, von einem Notar beglaubigt und beim Registergericht eingereicht werden. Die Vertretungs- und Geschäftsführungsbefugnisse verlangen eine Regelung, ebenso die Höhe der Einlagen, die Beteiligung am Stammkapital und am Gewinn der Gesellschaft. Die GmbH besteht erst mit der Eintragung in das Handelsregister. Dies dauert seine Zeit. Wer es eilig hat und die Briefbögen bereits drucken will, darf sich aber bereits GmbH in Gründung (i. G.) nennen, sobald der Notar seinen Siegel unter den Gesellschaftsvertrag gesetzt hat. Der Chef der GmbH führt den stolzen Titel Geschäftsführer. Es können auch mehrere Geschäftsführer bestellt werden.
Der oder die Geschäftsführer haben in der Regel das Recht, die Gesellschaft zu vertreten, das heißt, sie können im Namen und mit Wirkung für die GmbH Verträge abschließen und Verbindlichkeiten eingehen. Die GmbH muß sich einen sach- oder personenbezogenen Namen (Firma) geben und unterliegt den Vorschriften, welche das Handelsgesetzbuch für Vollkaufleute aufstellt. Für Anfänger ist sie damit kein geeigneter Einstieg. Spätestens beim Versuch, eine Eröffnungsbilanz aufzustellen, wird der kaufmännisch unerfahrene Anfänger das Handtuch werfen.

Alleingang

Wer unbedingt eine GmbH gründen will, aber keinen geeigneten Partner findet, kann dies auch allein tun und eine Ein-Mann-GmbH in die Welt setzen. Er muß das gesamte Stammkapital alleine aufbringen, kommt so aber in den Genuß der Haftungsbeschränkung und muß sich nicht mit anderen darum streiten, wer Geschäftsführer ist.
Das Steuerrecht bietet für die GmbH Vor- und Nachteile. Da ist zunächst die Gewerbesteuer. Während der Einzelkaufmann, die OHG oder die KG die Tätigkeitsvergütungen für ihre Gesellschafter nicht als gewerbesteuermindernd abziehen kann, darf die GmbH ihrem Geschäftsführer ein Gehalt auszahlen. Dieses Geschäftsführergehalt mindert die Gewerbesteuer. Die GmbH unterliegt als juristisch eigenständige Person der Körperschaftssteuer (Einkommenssteuer der juristischen Personen).
Erzielte Gewinne werden mit einem Körperschaftssteuersatz von 50 Prozent zur Versteuerung herangezogen. Gezahlte Gehälter an Geschäftsführer und Angestellte sowie Rückstellungen für Pensionen wirken sich gewinnmindernd aus. Werden die Gewinne an die Gesellschafter ausgeschüttet, ermäßigt sich der Körperschaftssteuersatz auf 36 Prozent. Die Gesellschafter müssen auf ihre Gewinnanteile Einkommenssteuer entrichten.
Die bereits von der GmbH entrichtete Körperschaftssteuer wird hierbei als Vorauszahlung auf die persönliche Einkommenssteuerschuld der einzelnen Gesellschafter angerechnet. Die an die Geschäftsführer gezahlten Gehälter unterliegen ebenfalls der Einkommenssteuerpflicht. Hierbei kommt der Geschäftsführer aber in den Genuß von steuerlichen Vorteilen wie Freibeträgen und abzugsfähigen Sonderausgaben, die jedem Arbeitnehmer zustehen.
Neben der Körperschaftssteuer unterliegt die GmbH der Kapitalertrags-, Vermögens- und Gesellschaftssteuer. Ob sich die Gründung daher in steuerlicher Hinsicht und im Vergleich zu anderen Gesellschaftsformen tatsächlich lohnt, sollte man am besten vorab mit dem Steuerberater genau durchrechnen.

Schatz im Silbersee

Die beste Idee und das passende Rechtskleid nützen wenig, wenn das nötige Kleingeld zum Aufbau des Unternehmens fehlt. Kosten in nicht zu unterschätzender Höhe entstehen in der Anlaufphase sowie in der Vorlaufphase der Gründung. Steuerberater, Rechtsanwalt, Notar und Unternehmensberater arbeiten ungern zum Nulltarif. Bei Anmeldungen zur Gewerbekartei und zum Handelsregister werden Gebühren fällig.
Die Beschaffung des Kapitals, Verhandlungen mit Banken und Kreditgebern nehmen viel Zeit und Mühe in Anspruch. Um die Kosten zu dämpfen, sollten alle Förderungsmöglichkeiten der öffentlichen Hand abgecheckt und Finanzquellen aufgespürt werden. Wer hier sich dieser Mühe nicht unterziehen will, verschenkt oft Geld. Einige der wichtigsten Förderungs- und Finanzierungsmöglichkeiten für Existenzgründer lesen Sie in dem Kasten auf Seite 29.

Der Autor ist Rechtsanwalt in Wuppertal

Nachdruck aus c't – Magazin für Computertechnik, Heft 6 und 7/1990

---

[1] Die Volkskammer hat das bereits übernommene GmbH-Gesetz der Bundesrepublik dahingehend abgeändert, daß die Mindesteinlage von 50 000 DM auf 20 000 DM reduziert wird. Mit der Gesetzesmodifikation soll die Gründung von Kleinunternehmen erleichtert werden. Die bis 1992 befristete Übergangsregelung gilt auch für bereits gegründete Gesellschaften mit beschränkter Haftung. Die zu der niedrigen Einlage gegründeten GmbH müssen diese bis 1995 auf 50 000 DM erhöhen. MP

Termine

Das Ostbayerische Technologie-Transfer-Institut (OTTI) führt 1990 etwa 100 Fachseminare, Kurse und Tagungen durch. Im Oktober/November werden folgende Fachseminare und Intensivkurse angeboten:

- **Programmiersprache C für Fortgeschrittene** — 29.-31.10. Regensburg und 5.-7.11. Weiden
- **Das relationale Datenbanksystem ORACLE - Grundkurs** — 5.-7.11. Regensburg
- **Programmierung in C++ - Einführung** — 13./14.11. Regensburg
- **Sicherheit von Rechenzentren und verteilten Informationssystemen** — 15./16.11. Regensburg

Ostbayerisches Technologie-Transfer-Institut e.V., D.-Martin-Luther-Straße 10, W-8400 Regensburg Tel. 0941-58273 (Herr E.Günther)

Der endgültige Compiler

Der KRS GmbH ist mit FCO (neuer deutscher Produktname!) die Plazierung einer echten Alternative für d-Sprachen-, Clipper™- und C-Programmierer im deutschen Markt gelungen. FCO ist Programmiersprache, Datenbank und Compiler in einem.

Die Schnelligkeit dieser **Wunderwaffe** wird durch die Generierung echter EXE-Files ohne Einbindung interpretativer Befehlsdateien erreicht. Daher ist FCO als professionelles Programmierwerkzeug zur Entwicklung von komplexen Datenbankapplikationen prädestiniert.

Anders als andere Compiler erzeugt FCO vollständigen Maschinencode und verzichtet auf speicherresidente Befehlsbibliotheken. Der Umfang der .EXE-Dateien wird bis zu 100x **kleiner und** die Geschwindigkeit **bis zu 20x größer.**

Wo gehobelt wird, fallen Späne. Hinter FCO steht ein vollkommen neues Compiler-Konzept, das sich am Sprachumfang von dBase-III orientiert. Aufgrund seiner Leistungsmerkmale sucht der Compiler im gesamten d-Sprachen-Bereich seinesgleichen. Vorhandene Applikationen können der Syntax von FCO angepaßt werden. C- und Assembler-Routinen können problemlos eingebunden werden.

In entscheidenden Punkten konnte die dBase-III-Plus-Syntax optimiert werden. Einige der vielen Vorteile sind: Die benutzerdefinierten Funktionen (UDFs) zur individuell unbegrenzten Erweiterung des FCO-Befehlssatzes. Der Syntax-Check zur gnadenlosen Aufdeckung nicht definierter Variablen beim Compilieren. Die Verwendung neuer numerischer Datentypen für größere Flexibilität und Geschwindigkeit.

FCO realisiert, worüber sich die Entwickler anderer Compiler noch den Kopf zerbrechen: Weg vom reinen dBase-Image. Hin zur Verfahrensweise konventioneller Compiler mit der dBase-Sprache als Basis.

Mit dem **Kampfpreis** von DM 1707,- inclusive Mwst rückt die KRS das Preis-/Leistungsverhältnis in der Compiler-Welt zurecht.

FCO

Voraussetzungen: 256 KB Arbeitsspeicher, MS-Dos ab Version 2.11, Festplatte
Exklusiv: KRS GmbH, D-02174/679-0

ZÜLOW-Software
Robert-Koch-Str. 14 • Templin, 2090
Tel.: 2821

Checkliste

Compiler:
- Erzeugt (als erster d-Sprachen-Compiler) vollständigen Maschinencode
- Schachtelbare GET...READ Anweisungen
- Präprozessor für bedingte Compilierung und Konstantendefinition
- Erweiterte Menü-Anweisungen

Programm:
- direkt ausführbare .EXE-Datei

Ausblick

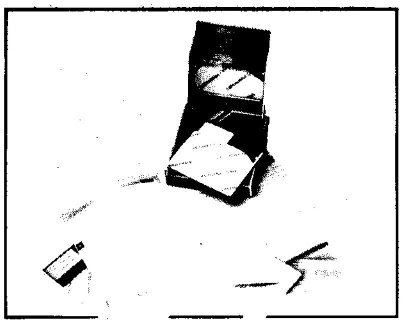
Professionelle Alternative zu vergleichbaren Produkten. Deutsche Version und Tools-Anpassung in Vorbereitung.

Nächster Messetermin

ORGATEC
INTERNATIONALE BÜROMESSE KÖLN
25.–30. OKT. 1990
Halle 1.2 O62/P61

Eingetragene Warenzeichen.: dBase v. Ashton Tate, Clipper v Nantucket

Den PC sicher im Griff- mit Christiani-Fernlehrgängen



PC-Anwendungspraxis

Zug um Zug lernen Sie in diesem Fernlehrgang die Möglichkeiten Ihres PC effektiv zu nutzen. Von Anfang an arbeiten Sie am Computer: keine Zeile Theorie bleibt ohne praktische Anwendung. Und die Programme liefern wir gleich mit. Alles wird Ihnen am praktischen Beispiel erklärt. So erleben Sie den PC-Einsatz "live".

12 reichhaltige, praxisbezogene Lehrbriefe.

4 ausgezeichnete Anwenderprogramme.

Viele Lern- und Wiederholungsdisketten.

Fundierte Studienbetreuung, begehrtes Abschlußzeugnis.

Nach diesem Lehrgang beherrschen Sie MS-DOS (bis 4.01), Textverarbeitung, Tabellenkalkulation, Datenbanksysteme und Geschäftsgraphik.

Fordern Sie gleich Informationsmaterial an:

Weitere Lehrgänge zu Programmiersprachen und PC-Anwendungen

PASCAL-Grundlagen

Einführungslehrgang in die Programmierung von Pascal.

BASIC-Grundlagen

Einführungslehrgang in BASIC allgemein.Strukturiertes Programmieren.

SPS-Programmierung

Einführung in Speicher-Programmierte-Steuerungen mit Simulationsprogramm. 5 Lehrbriefe.

Digital-Labor

Einführungslehrgang in die Bauelemente und Schaltungstechnik der Digitaltechnik mit Logik-Simulationsprogramm.

BASIC & Mikrocomputerpraxis

Intensivlehrgang BASIC mit reichhaltigen Anwendungsbeispielen. 14 Lehrbriefe.

Informationscoupon:

Senden Sie mir zu folgendem Lehrgang:

Lehrgangstitel

☐ Informationsmaterial

☐ Den 1.Lehrbrief kostenlos für 3 Wochen zum Test.

Adresse:

Name, Vorname

Straße, Nr.

PLZ, Ort 45249

Dr.-Ing. P. Christiani GmbH · Technisches Lehrinstitut und Verlag · 7750 Konstanz
Hermann-Hesse-Weg 2 · Telefon 07531-5801-0 · Telex 733 304 · Btx.: * 64748 #

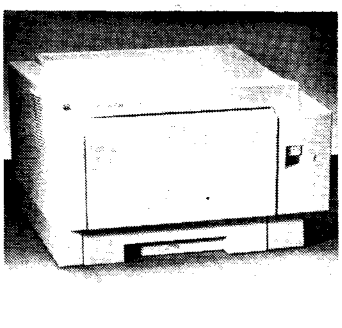
In Österreich: Ferntechnikum Bregenz · Belruptstraße 45 · 6901 Bregenz

Christiani

Osteuropäische Zeichensätze bald standardisiert?

Das Interesse westlicher PC-Hersteller am osteuropäischen Markt hielt sich in der Vergangenheit nicht zuletzt wegen der Beschränkungen der Cocom-Liste und des relativ kleinen Marktvolumens sehr in Grenzen. Als Folge baute jedes Land seine eigene PC-Industrie auf, die zwar teilweise auf westlicher Technologie basierte, aber in vielem – wie beispielsweise in den Definitionen der Zeichensätze – von internationalen Standards abwich. Microsoft und Ashton-Tate beriefen nun aus diesem Grunde einen Workshop ein, der sich dieser Problematik angenommen hat und so ein deutliches Wachstum der osteuropäischen PC-Industrie ermöglichen soll. Experten von IBM, Hewlett-Packard, Microsoft, Ashton-Tate, Lotus, Aldus und Cherry-Keyboards erörterten gemeinsam mit Spezialisten aus Bulgarien, Ungarn, Polen, Jugoslawien, der Sowjetunion und der Tschechoslowakei die Fragen eines einheitlichen Standardzeichensatzes und einigten sich auf eine für alle osteuropäischen Länder gültige Tabelle – die als „Latin 2" bezeichnete IBM-Zeichensatztabelle 852. Hierin sind alle auf dem lateinischen Alphabet basierenden Sonderzeichen enthalten – allerdings nicht die kyrillischen. Damit ist ein erster Schritt getan, um das in den neunziger Jahren zu erwartende Anwachsen der internationalen Datenkommunikation insbesondere im Handel und im Bankwesen zu ermöglichen. Neben der Firma Ashton-Tate, die „alle Übersetzungen in osteuropäische Sprachen auf der Basis dieses Zeichensatzes" ausführen will, verpflichteten sich auch Microsoft, Aldus und Lotus, ihre Produkte auf der Basis von Latin 2 zu entwickeln. Das betrifft insbesondere dBase IV, MS-Works und das DTP-Programm PageMaker. *MP*

Apple Personal LaserWriter



Als Angebot an die Apple-Einsteiger bietet die Firma jetzt zwei neue Laserdrucker im Leistungsbereich von 4 Seiten/Minute und 300 dpi an. Dabei ist das *Modell SC* mehr an den Bedürfnissen der individuellen Mac-Anwender ausgerichtet wie DTP, Korrespondenz, Berichte und Präsentationen. Der SC wird über den SCSI-Anschluß und die Quick-Draw-Software des Macintosh betrieben. Er hat einen 68000-Prozessor, 1 MByte RAM und vier Standardzeichensätze.

Das *Modell NT* dagegen kann von 2 bis auf 8 MByte RAM erweitert werden, besitzt schon in der Grundausstattung über 35 Standardschriften und arbeitet auf der Basis von Postscript. Es soll damit auch den Ansprüchen von Arbeitsgruppen gerecht werden. Da der Drucker mit allen LokalTalk-Netzfunktionen ausgestattet ist, kann er über eine Anschlußbox in einem Netz mit Mac-, Unix-, OS/2- und MS-DOS-Computern zusammenarbeiten.

Festplattenaufrüstung für Robotron-PCs

Von der Münchner CTT – Computertechnik und -technologie GmbH werden jetzt für den EC 1834 und EC 1835 Festplattenaufrüstsätze angeboten. Sie bestehen jeweils aus dem Festplattenadapter, einem Kabelsatz, einem 3,25-Zoll-Einbauwinkel, Schrauben, der Einbauanleitung und natürlich einer Festplatte. Im Angebot sind zwei Varianten mit 20 MByte (zu 375 DM) und 40 MByte (zu 499 DM, jeweils plus MwSt). Geworben wird für diese Lösung mit dem Slogan „100 % Robotron getestet und empfohlen".

Motorola mit schnellen SRAMs

Motorola produziert nicht nur die am häufigsten angewendeten dynamischen RAMs (DRAMs), sondern auch zwei Familien von statischen RAMs (SRAMs). Die DRAMs werden wegen ihres geringen Preises und ihres höheren Integrationsgrades bevorzugt in den Hauptspeichern von Rechnern eingesetzt. Motorola produziert Typen mit 256 KBit, 1 und 4 MBit. Weil die DRAMs aber langsamer als SRAMs sind, werden sie nicht in den schnellen Cache-Speichern moderner Mikrorechner verwendet.
Die Caches, die zur Beschleunigung des Speicherzugriffs zwischen die CPU und den Hauptspeicher geschaltet werden, benötigen immer schnellere Speicherbauelemente, da die Taktfrequenzen der Prozessoren ständig steigen – Prozessoren werden zur Zeit mit 33 MHz und mehr angeboten.
Die schnellen Standard-SRAMs heißen bei Motorola FSRAMs (fast-SRAMs), nicht zu verwechseln mit FRAMs (ferroelectric RAMs). Diese Standard-FSRAMs werden in verschiedene Varianten (16 bis 256 KBit) mit Zugriffszeiten von 12 bis 55 ns angeboten. Den 256-KBit-Schaltkreis MCM6207 beispielsweise gibt es mit 15, 20 und 25 ns.
Zusätzlich zu den Standard-FSRAMs, die keine Taktversorgung benötigen, wurden synchrone FSRAMs entwickelt. Sie besitzen unter anderem zusätzlich Latches für alle Ein- und Ausgänge sowie einen vereinfachten Schreibzyklus. Mit den synchronen FSRAMs läßt sich der Aufwand an externer Hardware senken, und der Speicher kann mit höheren Systemgeschwindigkeiten arbeiten. Der 256-KBit-Schaltkreis MCM 62482 beispielsweise kann mit Zugriffszeiten von 12 und 15 ns eingesetzt werden.

JESSI unterstützt Waferscale-Projekt

Den JESSI-Stempel hat jetzt das von der Anamartic Ltd., der Bull S. A. und Siemens getragene Projekt zur Entwicklung von Festkörper-Zwischenspeichern für Hochleistungscomputer erhalten. Dieser Speichertyp beruht auf einer die gesamte Siliziumscheibe umfassenden, als Waferscale bezeichneten Architektur.
Für Minicomputer und Workstations sind die üblichen magnetischen Hard- und Floppy-Disks häufig nicht schnell genug. Eine Lösung bietet die Waferscale-Technologie, mit der Festkörper-Zwischenspeicher realisiert werden können, die eine ganze Siliziumscheibe umfassen. Zur Herstellung der Mikrostrukturen werden die gleichen Verfahren wie bei dynamischen 4-MBit- und 16-MBit-Speichern verwendet. Speziell entwickelte Soft- und Hardware ermöglicht die Schaffung eines sich über die gesamte Siliziumscheibe erstreckenden Netzes von Speicherzellen, in dem automatisch defekte Bereiche erkannt und umgangen werden. Gegenüber den aus einzelnen Bausteinen aufgebauten Zwischenspeichern lassen sich so bei verbesserter Zuverlässigkeit bis zu 35 % Kosten sparen.

Datenkompression im Laserdrucker

Mit einem neuen Verfahren zur Speicheroptimierung rüstet Brother Industries, Nagoya/Japan, jetzt seine Palette von Laserdruckern aus; vom 4-Seiten-Laser HL-4 bis zum postscriptkompatiblen HL-8PS. Das von Brother entwickelte Datenkomprimierungsverfahren, Data Compression Technology (DACT) genannt, macht es möglich, mit Brother-Druckern Dateien bis zur mehrfachen Größe, je nach Aufbau der Datei, des eingebauten Speichers zu drucken. DACT komprimiert die vom PC eingehenden Datei-Informationen und lädt sie in den internen Speicher. Dabei werden nicht die einzelnen Bildpunkte in den Speicher eingelesen, vielmehr sorgt DACT für die Zusammenfassung gleichartiger Bildinformationen und erstellt einen Zugriffsalgorithmus für den korrekten Ausdruck der Datei. So ist es beispielsweise möglich, eine Grafik vom Format DIN A4 mit einer Auflösung von 300 dpi, die etwa 1 MByte Speicherkapazität benötigt, in einem Speicher von 512 KByte unterzubringen. Unabhängig davon können die Vorteile von DACT auch in Kombination mit Speichererweiterungen genutzt werden.

Tandon PCAs1/486

Einen 486er, bei dem verschiedene neue Konstruktionsideen verwirklicht wurden, präsentierte jetzt Tandon. Der PC wurde nicht als File-Server konzipiert, sondern für den Einstieg in Einzelplatzanwendungen mit zeit-, rechen- und speicherintensiven Aufgaben sowie Windows 3.0-Applikationen. Dank der hohen Integrationsdichte des von Tandon entwickelten MIAT-Chipsatzes konnte eine kompakte Bauweise erzielt und der PC in einem Slimlinegehäuse untergebracht werden. Eine ebenfalls von Tandon entwickelte Power-Poster-Logik soll gegenüber anderen PCs Leistungsvorteile bringen, indem sie die 32-Bit-Schreibanweisungen vom Prozessor ohne Wartezyklen entgegennimmt, Pipeline-Schreibanweisungen zum MIAT unterstützt und so den Prozessor für andere Aufgaben freistellt. Auch dank dieser Merkmale soll der PCAs1/486 eine Systemleistung von 10,3 MIPS erreichen.
In der Grundausstattung besitzt der neue Tandon 1 MByte Hauptspeicher (auf der Hauptplatine mit SIMMs bis zu 5 MByte, insgesamt bis zu 16 MByte aufrüstbar), eine 110-MByte-Festplatte sowie eine parallele und zwei serielle Schnittstellen. Trotz des kleinen Gehäuses stehen noch 3 AT-kompatible 16-Bit- und ein XT-kompatibler 8-Bit-Steckplatz zur Verfügung.

Keine Scanner mehr von Siemens?

Wie die PC-Woche im August meldete, beabsichtigt(e) die Firma, ab bzw. seit Ende September keine Scanner mehr zu produzieren. Obwohl die Marktprognosen für die Einlesegeräte günstig stehen, geht man bei Siemens nach den Absatzerfahrungen mit der Highscan-Familie der letzten zwei Jahre offenbar davon aus, daß der Scannermarkt noch nicht reif sei. Für die meisten Siemensianer kommt diese Entscheidung recht überraschend, da noch im Herbst neue Geräte auf den Markt kommen sollten. Der bereits angekündigte Farbscanner wird nun auch nicht mehr ausgeliefert werden.

Transceiver von AMD für kostengünstiges Ethernet

Advanced Micro Devices hat einen Transceiver-Baustein entwickelt, mit dem sich preisgünstige lokale Netze nach dem Ethernet-Standard aufbauen lassen. Verwendet werden keine dünnen Koaxialkabel wie beim Cheapernet, sondern zweiadrige, verdrillte Leitungen, wie sie von der Telefonie her bekannt sind. Der 10BASE-T-Transceiver-Baustein Am79C98 ist das Ergebnis eines vor etwa einem Jahr zwischen AMD und SynOptics abgeschlossenen Entwicklungsvertrages. AMD geht davon aus, daß 10BASE-T bis 1992 den größten Anteil am Ethernet-Markt haben wird.
Die meisten der bisher erhältlichen 10BASE-T-Karten sind im Transceiver-Bereich mit diskreten Bauelementen aufgebaut. Mit dem Einsatz der Am79C98 sollen sich die Kosten dafür um etwa 50 % reduzieren lassen. Außerdem ist der Am79C98 nach Angaben von AMD der einzige Baustein dieser Art, der über eine Power-down-Betriebsart verfügt. Das ist insbesondere bei der Anwendung in Laptops sehr hilfreich.
Der 10BASE-T-Baustein von AMD benötigt im Gegensatz zu ähnlichen Bausteinen weder einen externen Quarz noch einen Taktgenerator. Darüber hinaus verfügt er über eine

automatische Predistortion-Steuerung, eine automatische Polaritätserkennungs- und Korrektureinrichtung sowie eine Hilfsschaltung zum Anschluß eines Status-LED-Displays.
Der Am79C98 steht sowohl in einem 24poligen SKINNYDIP- als auch in einem 28poligen PLCC-Gehäuse zur Verfügung. Er läßt sich auch zusammen mit Ethernet-Controllern anderer Hersteller betreiben. *MP*

Marktentwicklung bei Rechentechnik

Nach Angaben von Diebold waren in der BRD am 1.1. 1990 3279815 Rechner installiert. Damit ist stückzahlmäßig ein gleichbleibendes Wachstum von 33,7 % erreicht worden. Wertmäßig konnte mit 10,9 % (auf 109,4 Mrd. DM) nicht an das Vorjahr (17,4 %) angeknüpft werden. Der Bereich kommerzielle Systeme verbuchte 1989 den letzten Zuwachs mit 35,1 %.
Der Marktführer IBM behauptete seine Position mit einem Zuwachs auf fast 500000 installierte Systeme. Bei den Systemen 3090 und 4381 gingen die Zuwachsraten deutlich zurück; auch die Installation des /3x-Systems war rückläufig. Enorme Steigerungsraten wurden beim AS/400 (2900 neu installierte Maschinen) und bei den PS/2-Systemen (144500 neu installierte Geräte) erzielt.
Siemens hat zusammen mit Nixdorf bei kleinen und mittleren Systemen die Spitzenposition erzielt. Das Stückzahlwachstum erreichte Siemens durch 215549 kleine und mittlere Systeme (insbesondere Sicomp und PCD). Bei großen Systemen sanken die Zuwachszahlen.
Die Firma Sun konnte 4000 Workstations und die Firma Silicon Graphics 561 Workstations verkaufen.
Olivetti konnte 20000 PCs installieren. DEC verdoppelte die Zahl der Installationen von VAX- und DEC-Stationen.
Insgesamt hat die Branche Büro-, Informations- und Kommunikationstechnik der BRD 1989 39,4 Mrd. DM umgesetzt. Gestiegene Einfuhren führten auf dem Gebiet der Kommunikations- und Informationstechnik zu einem Handelsdefizit. Erst im vierten Quartal 1989 erholte sich die Produktion. Schwächen im Hardware-Bereich wurden durch überproportionales Wachstum bei Software ausgeglichen. Besonders Anbieter von Komplettlösungen hatten gute Absatzchancen. Dienstleistungen und Software erzielten 1989 erstmals mit 35 Mrd. DM einen höheren Absatz als die inländische Hardware-Produktion. Für 1990 wird für die büro-, informations- und kommunikationstechnische Industrie einschließlich Software und Dienstleistungen ein Umsatzanstieg von 8 % erwartet.
Bei den Einfuhren dominierte Japan mit 46,2 %. Bei Telekommunikationsprodukten konnten japanische Firmen ihren Marktanteil in 4 Jahren verdreifachen.
Der Markt für professionelle PCs ist mit 31,1 % in der BRD im Jahre 1989 insgesamt überdurchschnittlich gewachsen. Nach Dataquest sollen 895000 PCs installiert worden sein. *Fa*

Neue Dimensionen in der Dateneingabe

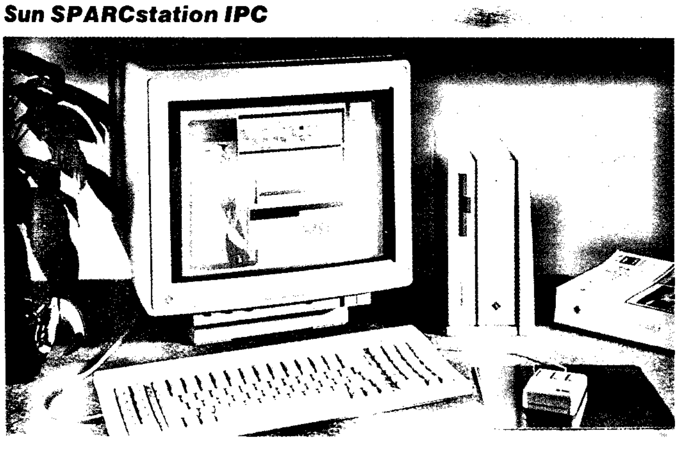
Ein neuartiges Eingabegerät, das eine Kombination aus intelligentem Grafiktablett mit den Funktionen einer Maus darstellt, wurde von der Firma CalComp GmbH unter der Bezeichnung **WIZ** vorgestellt. Mit diesem Eingabegerät sollen laut Hersteller aufwendige Befehlseingaben per Tastatur sowie das Durcharbeiten durch Menüstrukturen wegfallen. Der Bediener soll mit dem neuen Gerät in der Lage sein, in kürzester Zeit beispielsweise von der Textverarbeitung in ein Grafik-Softwarepaket einzusteigen. Das Eingabegerät WIZ benötigt im Gegensatz zur herkömmlichen Maus statt 102 nur 59 Klicks und mit Makroprogrammierung sogar nur 14 Klicks. Das entspricht einer Produktivitätssteigerung von zirka 80 Prozent. Außerdem soll das Gerät aufgrund seiner geringen Mechanik unempfindlich gegenüber Staubpartikeln sein. WIZ besitzt neben einem Fadenkreuz, mit dem das Zeichnen und das Übertragen von grafischen Layouts präzise ausgeführt werden kann, eine große Anzahl von Software-Wechselschablonen. Jede Schablone bildet die Befehlsoberfläche eines Softwarepakets ab. Die Befehle sind verschiedenfarbig kodiert und nach programmlogischen Gesichtspunkten zusammengefaßt, so daß der Anwender die Befehle auf einen Blick überschauen kann. Außerdem hat er die Möglichkeit, auf drei programmierbaren Benutzerfeldern der Wechselschablone längere Befehlsketten abzuspeichern und mit einem Klick wieder aufzurufen. WIZ kann fast mit jeder Standardsoftware (MS-DOS, MS-Windows) eingesetzt werden – auch mit Paketen, die keine Maus unterstützen. Gegenwärtig stehen für 35 verschiedene Programmpakete Software-Schablonen zur Verfügung.
Einsatzmöglichkeiten sieht die Firma CalComp vor allem in der Büroautomation, in Programmen der Textverarbeitung, bei Tabellenkalkulationen und bei Datenbanken, das heißt bei Aufgabenkomplexen, wo strenge Strukturen vorliegen. *Fa*

Elektromotor in Chip-Form

Das Massachusetts Institute of Technology (MIT) und die University of California haben im Rahmen eines Programms zur Entwicklung von Mikrostrukturen gemeinsam einen Mikrominiatur-Elektromotor entwickelt.
Der Rotor dieses Motors wird durch elektrostatische Felder bewegt und hat einen Durchmesser von 0,1 mm. Die vom Motor mit variabler Kapazität entwickelte „Antriebskraft" ist sehr gering. Probleme bestehen gegenwärtig noch mit der Ankopplung von Drehlasten an den Rotor, sowie bei größeren Chip-Motoren (mit praktisch verwertbaren Drehlasten) mit dem frei schwebenden Rotor.
Als gegenwärtige Anwendungsmöglichkeiten werden Sensoren oder Wandler gesehen. *Fa*

Als Ergänzung zu unserem Beitrag auf der 3. Umschlagseite stellen wir Ihnen im folgenden einige Workstations vor, deren Ankündigung uns nach Redaktionsschluß erreichte.

Sun SPARCstation IPC



Im Leistungssegment zwischen dem neuen Einsteigersystem SPARCstation SLC und dem High-End-System SPARCstation 1+ siedelte Sun jetzt den „Interpersonal Computer", IPC, an. Sun bezeichnet dieses Modell als die bisher preisgünstigste Farb-Workstation auf RISC-Basis. Der Einstiegspreis liegt bei etwa 22000 DM. In dem kompakten System sind in der Grundversion ein 3,5-Zoll-Floppylaufwerk, eine Ethernet-Schnittstelle und zwei serielle Schnittstellen sowie die von den anderen SPARCstations bekannten Audio-Ein-/Ausgabemöglichkeiten enthalten. Der 8-MByte-RAM kann mit 1- oder 4-MByte-SIMMs bis auf 24 MByte ausgebaut werden. Als Massenspeicher läßt sich eine 207-MByte-Festplatte integrieren, für Erweiterungskarten gibt es zwei Steckplätze mit dem S-Bus. Bei den Monitoren kann zwischen 16 Zoll color und 17 oder 19 Zoll monochrom gewählt werden. Als Anwenderzielgruppe sieht Sun insbesondere Industrie- und Dienstleistungsunternehmen im kommerziellen und technischen Bereich, die einen leistungsstarken Arbeitsplatzrechner im Netzverbund und mit Farbfähigkeiten suchen. Als Leistungsangaben werden von Sun 15,8 Dhrystone-MIPS/ 1,7 MFLOPS /10 SPECmark genannt.

Sony News 3140

Bereits seit einiger Zeit versucht Sony, auch im Workstationmarkt Anteile zu erringen. Als Basis der Geräte wird auf den RISC-Prozessor R 3000 der Firma MIPS zurückgegriffen, der bis zu 17 MIPS leistet. Bei dem jetzt angekündigten neuen Desktop-Modell 3140 ist der Koprozessor R 3010 integriert, mit dem 2,3 MFLOPS (Linpack) erreicht werden. Standardausstattung sind 8 MByte RAM, eine 240-MByte-Festplatte, eine Ethernet-, eine Audio-, eine Centronics- und zwei RS 232-Schnittstellen. Mit einem 19-Zoll-Monochrommonitor kostet eine solche Einstiegsversion knapp über 18000 DM.

Sony Portable News

Weiterhin kündigte Sony einen als Unix-Workstation bezeichneten portablen Rechner an. Mit 8 kg und 348 × 417 × 95 mm^3 bewegt man sich damit bereits in Richtung Laptop. Der Portable arbeitet mit dem Prozessor Motorola 68030 (25 MHz) und dem entsprechenden Koprozessor 68882 und leistet laut Sony knapp 4 MIPS (VAX-MIPS bzw. VUPs). Der 8-MByte-RAM kann bis zu 12 MByte erweitert werden. Integriert sind bereits eine 240-MByte-Festplatte, ein 3,5-Zoll-Floppylaufwerk und je eine parallele, serielle, Maus- und SCSI-Schnittstelle. Für eine Erweiterungssteckkarte ist im Gehäuse ebenfalls noch Platz. Als Betriebssystem wird News-OS, eine Implementation von BSD 4.3, verwendet; dazu X/Window V.11, NFS, XNS und TCP/IP. Der Preis soll bei etwa 26000 DM liegen.

SPARC LT AS1000/L10

Als welterster Laptop, der mit einer RISC-CPU als Hauptprozessor ausgestattet ist, wurde kürzlich der von Toshiba entwickelte Sparc LT präsentiert. Wie der Name verrät, wird der von Sun entwickelte Sparc-Prozessor verwendet, der dem Gerät eine Leistung von reichlich 13 MIPS verschafft. Weitere Merkmale sind 8 MByte RAM (maximal 40 MByte), 64 KByte Cache, eine 180-MByte-Festplatte, ein 3,5-Zoll-Floppylaufwerk und ein hintergrundbeleuchtetes 12-Zoll-Aktivmatrix-LC-Display (TFD) mit 1152 × 900 Pixel Auflösung. Für Erweiterungen gibt es 2 Steckplätze (1 × Sbus, 1 × Modem). Als Betriebssystem wird SunOS Unix R verwendet; wegen der Kompatibilität zu den Workstations mit Sparc-Prozessoren, zum Beispiel Suns SPARCstations, können etwa 1700 Softwarepakete genutzt werden. Als Anwendungsbereiche werden CAD/CAM, DTP und Software-Engineering genannt. Aufgrund der Masse (7,8 kg), der Maße (340 × 375 × 98 mm^3) und der (offenbar) fehlenden Unabhängigkeit vom Stromnetz wird der Sparc LT wohl weniger als Schoßcomputer denn als tragbare Workstation eingesetzt werden.

Computer-Club

Residentes Druckprogramm unter CP/M

Torsten Jäkel, Berlin

Das Betriebssystem CP/M ermöglicht prinzipiell nur die Abarbeitung jeweils eines Programms. Das Drucken einer Datei erfolgt dabei durch den Aufruf eines entsprechenden Programms, das nicht vor Abschluß des Drucks verlassen werden kann.

Im folgenden möchte ich Ihnen nun ein Programm vorstellen, mit dem die Ausgabe einer Datei an den Drucker im Hintergrund erfolgt und somit eine quasiparallele Programmabarbeitung unter CP/M ermöglicht wird.

Das Programm Spooler (Bild 1) wird wie jedes andere Anwenderprogramm aufgerufen und gestartet. Es besteht aber aus zwei Teilen, dem transienten, der das Starten des Druckprogramms (Spooler) übernimmt und dem residenten zum Drucken der Datei. Während des Druckens muß sich der residente Teil ständig im Hauptspeicher befinden und daher vor dem Überschreiben durch andere, nachfolgend zu startende Programme geschützt werden. Das erfolgt durch die Änderung der BDOS-Eintrittsadresse (**B**asic **D**isk **O**perating **S**ystem) in den Systemzellen 0006H und 0007H (siehe auch /1/ und /2/). Da durch einen Warmstart (<Ctrl>-<C>) oder durch das Beenden eines Programms die BDOS-Zeiger zurückgesetzt werden, muß das Druckprogramm stets neu resident gemacht werden. Dazu wird ein Sprung, der vom Betriebssystem nicht restauriert wird, direkt im BDOS modifiziert.

Parallele Programmabarbeitung

Das scheinbar parallele Laufen des Druckprogramms im Hintergrund eines anderen Programms wird durch die faktische Erweiterung des BDOS möglich.

Alle BDOS-Aufrufe werden zuerst zum Druckprogramm umgeleitet, das die Ausgabe der Daten übernimmt und anschließend den Systemruf an das eigentliche BDOS weitergibt.

Ein BDOS-Ruf hat also das Einschieben des Druckprogramms zur Folge, und das Drucken erfolgt scheinbar parallel.

Die Registerbelegung wird gesichert und spezielle BDOS-Funktionen überwacht, beispielsweise das Setzen der DMA-Adresse. Da diese Funktionen sowohl von einem Anwenderprogramm als auch vom Druckprogramm aufgerufen werden, kann es sonst zu Konflikten kommen.

Um den Zugriff auf die Diskette zu optimieren, wird nicht mit der logischen Sektorlänge 128 Byte, sondern mit der physischen (z. B. 1024 Byte/Sektor) gearbeitet. Damit werden zeitaufwendige Diskettenzugriffe verringert. Das Druckprogramm besitzt einen eigenen Puffer, der mindestens einen physischen Sektor der Diskette aufnehmen kann.

Installation

Das Programm wird durch den Assembler M80 und den Linker L80 generiert. Einzige anzupassende Konstante ist die Anfangsadresse (CCPANF) des CCP (Console Command Processor), die ermittelt werden muß. Dazu kann der Inhalt der Systemzellen 0006H und 0007H ausgewertet werden. Darin ist die Anfangsadresse des BDOS angegeben. Davon ist die Länge des CCP (2 KByte) abzuziehen. Eine andere Möglichkeit ist der

```
;Spooler für CP/M 2.2
;(C) Torsten Jäkel, 25.5.90
;abhängiger Installationswert !:
CCPANF   equ     0c800H              ;Anfang CCP
;
BDOS     equ     00005h              ;BDOS-Ruf
DEFFCB   equ     0005Ch              ;Standard FCB
mustlen  equ     7                   ;Länge des Musters
         .z80
;testen, ob Spooler aktiv ist
anfang:  ld      hl,kenn1
         ld      de,kenn2
         ld      b,mustlen
pruef:   ld      a,(de)
         cp      (hl)
         jr      nz,inaktiv
         inc     hl
         inc     de
         djnz    pruef
;Spooler noch aktiv
         ld      de,msg1
outerr:  ld      c,09h
         call    BDOS                ;Fehlerausgabe
         jp      0000h               ;Warmstart
;wenn Open-Fehler
error:   ld      hl,kenn2
         ld      b,mustlen
cl_sp:   ld      (hl),0
         inc     hl                  ;Kennung weg
         djnz    cl_sp
         jr      outerr
;Fehlertexte
msg1:    DB "spooler activ"
         DW      0a0dh
         DB      24h
msg2:    DB "can't open"
         DW      0a0dh
         DB      24h
msg3:    DB "no file"
         DW      0a0dh
         DB      24h
kenn1:   DB "SPOOLER"                ;Vergleichsmuster
;------ Spooler aktivieren -----
inaktiv::
         ld      hl,xx1
         ld      de,spooler
         ld      bc,xx3-spooler
         ldir
;testen, ob Datei angegeben
         ld      a,(DEFFCB+1)
         cp      ' '
         jr      nz,datist
         ld      de,msg3
         jr      error
;LW der Datei ermitteln
datist:  ld      a,(DEFFCB)
         and     a
         jr      nz,is_lw            ;mit LW angegeben
;akt. LW holen
         ld      c,19h
         call    bdos
         inc     a                   ;ab LW A=1 im FCB
         ld      (DEFFCB),a          ;ablegen
;FCB vorbereiten
is_lw:   ld      hl,DEFFCB
         ld      de,sp_fcb
         push    de
         ld      bc,16               ;Lange Standard-FCB
         ldir                        ;kopieren
         pop     de
;File eröffnen
         ld      c,0fh
         call    BDOS
         cp      0ffh
         jr      nz,no_err           ;Open fehlerhaft
         ld      de,msg2
         jr      error
;BDOS-Eintritt verbiegen
no_err:  ld      hl,(BDOS+1)         ;wo BDOS-Entry
         ld      (BDOSENT),hl        ;alter BDOS-Anfang
         inc     hl
         ld      e,(hl)
         inc     hl
         ld      d,(hl)              ;interner BDOS-JP
         ld      (BDOSDI+1),de       ;Adresse wohin merken
         ld      de,spooler
         ld      (hl),d
         dec     hl
         ld      (hl),e              ;zum Spooler
;Okay, Spooler aktiv
         jp      0000h               ;Warmstart
;------ residenter Spooler -----
xx1:     .phase  CCPANF-0600h
spooler:jp       weiter
bd_err: DEFS     8                   ;BDOS-DW-Leiste
;Vergl.muster
kenn2:   DB "SPOOLER"
;
weiter:  di                          ;keine INT möglich
         ld      (sprett),sp
         ld      sp,STACK            ;Spooler-Stack
         push    hl
         push    de
         push    bc
         push    af                  ;Register retten
;Spooler jedesmal resident machen
         ld      hl,spooler
         ld      (BDOS+1),hl
;BDOS-Rufe testen
         ei
         ld      a,1ah               ;set DMA ?
         cp      c
         jr      nz,weil
         ld      (DMA),de            ;merken
;sind noch Daten da ?
weil:    ld      hl,(offset)
         ld      a,h
         or      l
         jr      nz,zei_da           ;noch da
;Sektoren einlesen
         ld      de,secpuff
         ld      b,8                 ;8 mal 128 Bytes
lessect:push     bc
         push    de
         ld      c,1ah               ;DMA setzen
         call    BDOSDI              ;direkt anspringen
         ld      de,sp_fcb
         ld      c,14h               ;sequenziell lesen
         call    BDOSDI
         pop     hl                  ;aktuelle DMA-Adresse
```

```
        pop     bc              ;Sektoranzahl
        and     a
        jr      nz,datend       ;Dateiende ?
        ld      de,128
        add     hl,de
        ex      de,hl
        djnz    lessect
;letzten Sektor gelesen
datend: ld      a,b
        cp      8               ;kein Sektor ?
        jr      z,spentf
;Drucker bereit ?
zei_da: ld      hl,(0001h)      ;BIOS-WBOOT
        ld      de,14*3
        add     hl,de           ;Adresse BIOS-Listst.
        ld      de,retadr
        push    de
        jp      (hl)            ;wie CALL
retadr: and     a
        jr      z,zubdos        ;nicht bereit
;Zeichen holen
        ld      de,(offset)
        ld      hl,secpuff
        add     hl,de           ;weiter
        ld      a,(hl)
        ld      e,a
        cp      1aH             ;<Ctrl> <Z> = Textende
        jr      z,spentf
        cp      0dh
        jr      nz,ausg1
        xor     a
        ld      (c_count),a     ;Anfang Zeile
        jr      ausg3
ausg1:  cp      09h             ;Tabulator ?
        jr      nz,ausg2
        call    tabul           ;Leerzeichen ausgeben
        jr      ausg4
ausg2:  cp      0ah
        jr      z,ausg3         ;LF nicht mitzählen
        ld      a,(c_count)
        inc     a
        ld      (c_count),a     ;Zeichen zählen
ausg3:  ld      c,05h           ;Druckerausgabe
        call    BDOSDI
;Puffer zu Ende ?
ausg4:  ld      hl,(offset)
        inc     hl
        ld      (offset),hl
        ld      de,8*128        ;Puffergröße
        and     a
        sbc     hl,de
        jr      nz,zubdos
        ld      hl,0            ;Puffer leer
        ld      (offset),hl
;ausgegeben, jetzt gerufenes BDOS
zubdos: ld      de,(DMA)
        ld      c,1ah
        call    BDOSDI          ;reset DMA
        di
        pop     af
        pop     bc
        pop     de
        pop     hl
        ld      sp,(sprett)
        ei
        jp      BDOSDI
;Datei fertig, Spooler entfernen
spentf: ld      de,sp_fcb
        ld      c,10h           ;close
        call    BDOSDI
        ld      de,msg4
        ld      c,09h
        call    BDOSDI          ;Nachricht
        di
        ld      hl,(BDOSENT)    ;alter BDOS-Anfang
        ld      (BDOS+1),hl     ;entfernen
        ld      de,(BDOSDI+1)
        inc     hl
        ld      (hl),e
        inc     hl
        ld      (hl),d          ;BDOS reparieren
        ld      hl,kenn2
        ld      b,mustlen
clear:  ld      (hl),0
        inc     hl
        djnz    clear           ;Kennung weg
        jr      zubdos
;
tabul:  ld      c,05h           ;BDOS-Nr. 5
        ld      e,' '
        call    BDOSDI          ;Leerzeichen
        ld      a,(c_count)
        inc     a
        ld      (c_count),a     ;Zeichenzähler +1
        and     07h
        ret     z               ;Position erreicht ?
        jr      tabul
;
msg4:   DW      0a0dh
        DB "spooler ready  "
        DW      0a0dh
        DB      24h
;
;RAM-Zellen:
sprett: DW      0
DMA:    DW      0
offset: DW      0               ;keine Bytes da
c_count:DB      0               ;Zeichenzähler
;BDOS direkt intern anspringen
BDOSDI:: jp     0
BDOSENT::DW     0               ;alte BDOS-Adr.
;FCB für zu druckende Datei
sp_fcb: DEFS    35,0            ;mit 0 initial.
secpuff:DEFS    8*128           ;Sektorpuffer
;Stack des Spoolers
        DEFS    20
STACK:
xx3::
        .dephase
        end
```

Bild 1 Das Programm Spooler druckt im Hintergrund

Aufruf des Kommandos LOG im Programm Power /3/. In der letzten Zeile erhält man die letzte freie TPA-Adresse, die hier bereits die Adresse direkt vor dem CCP-Anfang ist.

Das Programm ist sehr einfach gehalten, und es ist nicht verschieblich. Es kann nicht gleichzeitig mit anderen speicherresidenten Programmen angewendet werden.

Während des Druckens von Dateien können andere Programme und Kommandos aufgerufen werden. Durch die Reservierung des Speicherplatzes für das Programm und die Ablage vor dem CCP stehen etwa 3,5 KByte weniger Speicherplatz für Anwenderprogramme zur Verfügung.

Das Programm druckt allerdings nur dann zyklisch, wenn auch kontinuierlich BDOS-Rufe erfolgen. Wird vom CCP ein neues Kommando eingelesen, aber nicht eingegeben, wird das Drucken unterbrochen. Dieser Zustand kann durch Aufruf des Programmes Power verhindert werden.

Das Programm Spooler leistet bereits in der vorliegenden Version gute Dienste, kann aber auch als Anregung für weitere eigene Lösungen dienen, die eine scheinbare parallele Abarbeitung unter CP/M erfordern, beispielsweise zur Meßwerterfassung oder zur Datenübertragung zwischen CP/M-Rechnern.

⊠ KONTAKT ☏

Humboldt-Universität zu Berlin, Sektion Elektronik, Invalidenstraße 110, Berlin, 1040; Tel. 2 80 35 95

Literatur

/1/ Geiler, J.: Nachladbare Gerätetreiber für Personal- und Bürocomputer. Mikroprozessortechnik, Berlin 1 (1987) 6, S. 179

/2/ Licher, E.; Massenbach, T.: CP/M 2 lernt dazu. c't, Hannover 4 (1987) 1, S. 87

/3/ Bernert, J.; Burow, M.; Hanisch, C.: CP/M in der Praxis. VEB Verlag Technik, Berlin 1988

Zitiertes

CHIP erreicht ... noch immer eine Flut von Bittschriften, ja Bettelbriefen. Klar: Hieß es doch noch in der CHIP-Ausgabe vom März 1990 auf Seite 17: „Wer Computer in die DDR spenden will – CHIP vermittelt Kontakte". Heute können wir mit Fug und Recht behaupten, die schlimmste (Computer-)Not sei vorbei. Trotzdem stehen DDR-Bürger weiterhin auf der Matte: „Zu meinem Glück fehlt mit eigentlich nur noch ein 386er-AT. Besorgen Sie den bitte zu einem symbolischen Kaufpreis!" Eine CHIP-Mitarbeiterin kommentierte diese Zuschrift trokken: „Ein AT fehlt auch mir zu meinem Glück. Bloß, wo soll ich betteln?" ... Wenn ich einen Rechner haben will, dann kaufe ich mir einen. Wenn es für einen großen Computer nicht reicht, spare ich eine Zeitlang oder begnüge mich mit einem kleinen. ... Und ich meine, für die vielen Anwender in der DDR sollte dieser Grundsatz nicht minder gelten.
aus CHIP 8/90

Dr. Peter Löber von der Technischen Universität in Chemnitz freut sich über den neuen Escom-XT für sein Institut. Die CHIP-Redaktion hatte ihm das Gerät geschenkt als Anerkennung für seine Bemühungen beim Beschaffen von fünf DDR-Computern, die im Labor in München den üblichen Tests unterworfen wurden.
aus CHIP 7/90

Wirkung und Ursache?

TERMIN

HIGH TEACH '90
Fachmesse für Bildung und Kommunikation

WER? Dr. Anton & Partner GmbH und Kammer der Technik
WANN? 26. bis 29. November 1990
WO? Cottus
WAS?
- vorschulische Erziehung und schulische Bildung
- berufliche Aus- und Weiterbildung
- allgemeine Weiterbildung
- Fach-/Hochschulstudium

WIE? Informationen über Dr. Anton & Partner GmbH, Frankfurter Straße 26, W-6453 Seligenstadt; Tel. (0 61 82) 2 70 98/99

Systemtuning mit TSR-Programmen: Grundlagen, Tips und Utilities zum Programmieren residenter Programme

von G. Born, Addison-Wesley, Bonn 1989, 250 S. incl. Programmdiskette, 69,– DM

Falls Sie mit dem Titel des Buches nichts anzufangen wissen: TSR (= Terminate and Stay Resident) steht als Kürzel für residente, also solche Programme, die im Speicher verbleiben und entweder als Systemerweiterungen unbemerkt ihre Arbeit verrichten (wie Append und Print) oder per Hotkey aktiviert werden (zumeist Utilities wie SideKick oder Online-Hilfen wie die Norton-Guides). Jeder DOS-Nutzer kommt also damit in Berührung. Weniger bekannt dagegen ist, wie solche TSRs programmiert werden, vor allem, wie Behinderungen zu vermeiden sind.
Ein Buch zu dieser Thematik ist nur zu begrüßen, denn Informationen sind rar. Das liegt vor allem daran, daß DOS ursprünglich nicht für das Nebeneinander mehrerer Programme gedacht war und eine Reihe nachträglicher Erweiterungen des Herstellers nur für den eigenen Gebrauch bestimmt sind. Systemfunktionen wie der Interrupt 28h oder das DOS-Critical-Region-Flag gelten – obwohl in vielen Programmen verwendet – als undokumentiert und tauchen deshalb in den Handbüchern nicht auf.
Zumeist müssen derlei Informationen aus den Nebensätzen der Profiliteratur und Fachzeitschriften zusammengesucht werden. Genau hier setzt das Buch ein. Neben der Vollständigkeit (sofern das überhaupt beurteilt werden kann) gefällt vor allem die übersichtliche Darstellung und gute Aufbereitung des Stoffes für den Leser. Deshalb ist das Werk gerade für den Einstieg in die Thematik geeignet.
Wenn Sie sich bereits eingehend mit speicherresidenten Programmen befaßt haben – erwarten Sie nichts Neues. Die recht vollständige Zusammenfassung und Systematisierung der diversen Möglichkeiten sucht aber ihresgleichen und ist so sicher auch zum Nachschlagen gedacht. Dagegen wurde auf einen Referenzteil fast völlig verzichtet. Es kann zwar vorausgesetzt werden, daß der fortgeschrittene Programmierer über Literatur zu den DOS-Interna verfügt, doch müssen sie beim Studium des Buches stets zur Hand sein.
Die zahlreichen Beispiele werden in Turbo-Pascal vorgestellt, ein paar kurze Programmsequenzen auch in Assembler. Das Lieblingsprogramm des Autors scheint der Crossreferenzlister XREF für Pascal zu sein. Nicht weniger als 38 Seiten nehmen die Verweise der kurzen und zudem auf Diskette vorhandenen Demoprogramme ein. Seitenfüller? Thematisch sind die Beispiele dagegen gut gewählt, und eine Reihe von ihnen kann ohne Veränderungen eingesetzt werden. Im einzelnen handelt es sich um einen Druckspooler, eine Systemuhr, ein Programm zum Abspeichern des Bildschirms sowie Tools zur Verwaltung residenter Programme (Anzeigen, Markieren, Freigeben). Daneben gibt es auch Informationen zu Themen, die nicht direkt mit TSRs zusammenhängen, etwa die Funktionsweise der EMS-Verwaltung.
Alles in allem ein Werk, das praxisnah Wissen vermittelt, dessen eigentlicher Wert aber im Zusammentragen der Erfahrungen mit residenten Programmen unter DOS besteht.

U. Schulze

Das Addison-Wesley Werk zu WORD 5.0

von Rainer Haselier, Addison-Wesley, Bonn 1990, 800 S., 1 Diskette, 68,– DM, ISBN 3-89319-180-1

Neben der Tatsache, daß dieses Buch schon allein durch einen Rechtsstreit zweier Verlage die Aufmerksamkeit auf sich zieht, hebt es sich wohltuend aus der Palette sonstiger Literatur zu WORD dadurch hervor, daß es für Einsteiger und Textverarbeitungsprofis gleichermaßen gedacht ist. Auf den Rechtsstreit deutet das nunmehr geschwärzte WORD „Standard" hin – auf den universellen Anspruch eines Lern- und Nachschlagewerkes der Umfang von 800 Seiten, womit dieses Buch dem Originalhandbuch von Microsoft in diesem Punkt in nichts nachsteht. Nach einem kurzen Blick wird auch sofort klar, daß es in manch anderer Hinsicht nicht nur die Originalliteratur ergänzt, sondern auch ersetzen kann – was dies auch immer hinsichtlich der Notwendigkeit zur lizensierten Benutzung von WORD bedeutet.
Der Einsteiger wird in diesem Buch systematisch in die Textverarbeitung eingeführt, wobei zum Studium nicht notwendigerweise der Computer hinzugezogen werden muß, da alle wesentlichen Bildschirm-Menüs abgebildet sind. Der Routinier wird seine Aufmerksamkeit auf die letzten Abschnitte des Buches richten, die sich mit spezielleren Themen wie Grafikeinbindung, Druckertreiber usw. beschäftigen und dem Anspruch eines Nachschlagewerkes vollauf genügen. Die Darstellungsweise ist insbesondere auf den ersten Seiten ausführlichst. Jede Tastenbetätigung und Tastenkombination wird unmißverständlich beschrieben. Diese Ausführlichkeit ermöglicht in erster Linie dem Nicht-DV-Fachmann den Einstieg – der Routinier wird dies sicher nachsehen. Die Präzision der Beschreibung der WORD-Bedienung steht aber in Widerspruch zur Qualität und Quantität der Hinweise zur Installation und Organisation der Textverarbeitung am PC. Dieses Problem wird eher formal abgehandelt. Zur professionellen Textverarbeitung gehört neben dem Schreiben von Texten auch deren Sicherung und Ablage unter Zuhilfenahme der Mechanismen des Betriebssystems. Wo zum Beispiel legt man die Textdateien vorteilhafterweise ab? – Sicher nicht im WORD-Directory, wie es bei einigen Beispielen offensichtlich wird. Gerade der Anfänger benötigt hier Hilfe. Der Profi wird dies weniger vermissen – ihn dürften die Ausführungen, die er zu den Druckertreibern und zur professionellen Textgestaltung findet, besonders erfreuen. Vielleicht hätte man sich hier noch ausführlichere Hinweise in Richtung des DTP gewünscht, was aber sicher den Rahmen des Buches sprengen würde. So ist das Buch ein Beispiel für einen Kompromiß, der es tatsächlich – ungeachtet aller Rechtsstreitigkeit um das ehemalige Wort „Standard" im Buchtitel – inhaltlich zu einem Standardwerk werden läßt.

Ch. Hanisch

Erfolgreiche Musterlösungen für PC-Host-Verbindungen

von G. Sollfrank, WEKA-Fachverlage GmbH Kissing 1989, 1900 Seiten, 2 Bände, 292,– DM

Der Autor stellt in seinem Werk praxiserprobte Hard- und Softwaregrundkonzepte für PC-Host-Verbindungen aus herstellerneutraler Sicht vor. Dabei werden im konzeptionellen Teil intelligente Workstations, Integrations-, und Netzwerkkonzepte sowie PC-Host-Verbindungen in lokalen Netzen ausführlich behandelt. Daran schließen sich Grundlagen der praktischen Rechnerkommunikation, Übersichten über Standardsoftware und Methoden zur Fehlersuche in den PC-Host-Verbindungen an. Protokolle und Dienste für PC-Host-Systeme, z. B. OSI-Referenzmodell, XMODEM-Protokoll, SNA, DECnet usw., werden detailliert praxisbezogen erläutert. Im praktischen Teil des Werkes werden PC-Host-Musterlösungen, so z. B. Terminalemulation, permanente Kommunikation, File-Transfer mit intelligenten Workstations, verteilte PPS-Systeme sowie externe und verteilte Datenbanksysteme (ORACLE) in Verbindung mit LAN-Konzepten, dargestellt. Abschließend werden Mittel und Methoden zur effektiven Vorbereitung, Schulung und Betreuung von PC-Host-Verbindungssystemen (z. B. Information-Center-Konzept, Profit-Center-Konzept) in der kooperativen Zusammenarbeit zwischen Organisations-/Informationsbereichen und den Anwendern in betrieblichen Fachabteilungen in Form von praxisnahen Modelldarstellungen vorgestellt.
Der Autor hat ein Übersichtswerk geschaffen, das jungen und praxiserfahrenen Informatikern, Organisatoren, aber auch Managern und Anwendern Unterstützung für ihre effiziente praktische Arbeit geben wird.

W. Schölling

TURBO PASCAL 5.5
Der Einstieg in 20 Schritten

von M.-A. Beisecker und P. Brickwede, SYBEX-Verlag, Düsseldorf 1989, 138 S., 19,80 DM, ISBN 3-88745-732-3

Das Büchlein der QuickStart-Buch-Reihe will – wie seine Vorgänger – dem Nutzer eines Programms ermöglichen, sich schnell mit diesem vertraut zu machen und ihn über dessen wesentliche Eigenschaften zu informieren. In 20 Schritten wird das Leistungsvermögen des Programms dargestellt, sowie eine grundlegende Einführung und Übersicht gegeben. Durch die Verwendung von „icons" lassen sich QuickStart-Bücher auch als schnell handhabbare Nachschlagewerke nutzen.
Die Autoren geben eine stringente und methodisch durchdachte Einführung in Turbo-Pascal 5.5, wobei die einzelnen Schritte logisch aufeinander aufbauen. Vor jedem Schritt ist – als Orientierungshilfe – die Zeit angegeben, die man in etwa benötigt, um den Schritt am Rechner nachzuvollziehen. Es ist aber auch möglich, die Schritte ohne Rechner abzuarbeiten. Den Anhang bildet ein ca. 20 Seiten langes Aktienprogramm. An Hand dieses Programms kann der Leser die einzelnen Schritte am Beispiel vertiefen. Gleichzeitig wird ein sauber strukturiertes Pascal-Programm vorgeführt, durch das die Blockorientiertheit der Sprache demonstriert wird.
Die Schritte 1 bis 15 befassen sich mit den Grundlagen der Programmierung in Turbo-Pascal 5.5.
In den Schritten 16 bis 20 werden der Debugger, die objektorientierte Programmierung, der EMS-Speicher sowie die Tool-Boxen für Turbo-Pascal abgehandelt. Entsprechend der Zeitvorgabe der Autoren benötigt man für die Durcharbeitung des Buches am Rechner etwa 10,5 Stunden.
Zusammenfassend kann eingeschätzt werden, daß mit TURBO-PASCAL 5.5 eine Publikation vorliegt, die insbesondere dem profilierten Turbo-Pascal-Programmierer den problemlosen Umstieg auf die Version 5.5 ermöglicht. Dem Pascal-Einsteiger sei das parallele Studium von Literatur zu Standard Pascal anempfohlen. Der Zeitaufwand wird freilich wesentlich größer als der des Umsteigers sein, aber auch für ihn wird er sich stark reduzieren.

C. Paul

Aktuelle EDV-Nachschlagewerke

Von der WEKA-Fachverlage GmbH sind zwei bemerkenswerte, umfangreiche Nachschlagewerke für die Organisations- und EDV-Arbeit in Unternehmen – insbesondere zur Verbesserung der EDV-Kompetenz und -Kenntnisse in kaufmännischen und technischen Unternehmensbereichen – herausgegeben worden:
1. Mertin, Carl-Otto: Aktuelle EDV-Musterpflichtenhefte zur präzisen Definition der kompletten Anforderungen an EDV-Lösungen in allen käufmännischen Bereichen. Ordner A5-Format, 1300 Seiten
2. Aktuelle EDV-Musterpflichtenhefte für alle technischen Unternehmensbereiche. Ordner A5-Format, 1400 Seiten.
Die beiden Nachschlagewerke beinhalten sehr detaillierte Musterpflichtenhefte für alle relevanten kaufmännischen und technischen Unternehmensaufgaben sowie Checklisten zur Beschreibung der Aufgaben, Aufnahme von Datenmengengerüsten und Projektplanung in einer für die praktische Arbeit notwendigen effektiven Darstellungsform.

Sch.

vorgestellt

Neue Workstations

Glaubt man den Berichten, so wird IBM mit dem zweiten Anlauf im Workstationmarkt ein großer Sprung gelingen: Die für dieses Jahr geplante Produktion von Modellen des neuen RISC-Systems/6000 soll längst verkauft sein, und auch in die etwas weitere Zukunft blickt man bei IBM zuversichtlich. Laut einer Umfrage nach den bei DV-Managern gefragtesten Workstations kam IBM mit seinem neuen Produkt aus dem Stand auf Platz zwei hinter der Sparcstation der Firma Sun und vor den Firmen HP/Apollo und DEC.

Das System/6000 hatten wir in unserem CeBIT-Bericht bereits vorgestellt (siehe MP 8/90, S. 33); hier wollen wir einige der Systeme nennen, die der Wettbewerb ins Rennen schickt.

Da wäre zunächst die offenbar als Preisbrecher bei Einstiegsmodellen gedachte Workstation von Sun mit Namen **SLC** – Super Low Cost. Das System soll für unter 12000 – exakt 11980 DM – angeboten werden (Bild 1). Allerdings handelt es sich dabei um eine sogenannte Diskless-Workstation, bei der weder Diskettennoch Festplattenlaufwerk integriert sind. Der Einsatz ist also vor allem als Workgroup Computer im Lokalen Netz zur Dokumentenbearbeitung, für das Software-Engineering, die Zeichnungserstellung beim CAD und bei Datenbankanwendungen gedacht. Jedoch kann der Arbeitsplatz über einen SCSI-Port auch mit externen Floppy- oder Harddisks und CD-ROM-Drives für den autonomen Betrieb versehen werden. Besonderes Merkmal ist die Unterbringung des SPARC-Prozessors (20 MHz; Leistung 12,5 MIPS, 1,25 MFLOPS, 7,6 SPECmark), des Hauptspeichers (8 bis 16 MByte), des Platten- und Netzwerkinterfaces, des Audiosystems sowie der Stromversorgung im 17-Zoll-Monochrom-Monitor (1152 x 900 Pixel). Doch nicht nur mit dem damit möglichen Verzicht auf ein Grundgerätegehäuse ist der SLC besonders anwenderfreundlich, sondern auch mit dem Wegfall des Betriebsgeräusches, da kein Lüfter notwendig ist.

Ebenfalls im Mai kündigte Sun die Weiterentwicklung der SPARCstation 1 an, die mit 80000 ausgelieferten Systemen und einem Weltmarktanteil von 54% die weltweit erfolgreichste RISC-Workstation sein soll. Die **SPARCstation 1+** hat bei 25 MHz Taktfrequenz eine Leistung von 15,75 MIPS, einen Grafikbeschleuniger, Farb- oder Monochrom-Monitor und integrierte Plattenlaufwerke.

Kurz nach der CeBIT kam im April von Digital Equipment die Kampfansage an IBMs System/6000 mit der Ankündigung der neuen Grafikworkstation **DECstation 5000, Modell 200** (Bild 2) als nunmehr leistungsfähigstes RISC-System der Produktpalette. (Mit dieser Ankündigung senkte die Firma übrigens gleichzeitig die Preise für die bisherigen RISC-Workstations: DECstation 2100 nunmehr 15000 DM, DECstation 3100 jetzt ab 22500 DM.) Auch die neue DECstation ist ein Diskless-System; jedoch lassen sich als Zusatzspeicher Festplattenlaufwerke, Streamer, DAT-Laufwerk und CD-ROM-Drive in der Desktop-Erweiterungsbox BA42 unterbringen. Herz der Rechner ist der CPU- und FPU-Chipsatz R3000/R3010 mit 25 MHz. Als Rechenleistungen werden 24 MIPS, 3,7 MFLOPS Linpack bei doppelter Genauigkeit und 18,5 SPECmark angegeben. Das System wurde insbesondere für rechenintensive 2D- und 3D-Grafikanwendungen entwikkelt und ist in vier Grafikkonfigurationen lieferbar: *Modell 200 CX* als Einstiegsversion, besonders für CASE-Anwendungen, ohne anspruchsvolle Grafik; *Modell 200 PX* mit schneller 2D-Vektor- und Raster-Grafikleistung für ECA und MCAD (mit neuem Pixelprozessor Pixelstamp); *Modell 200 PXG* mit 3D-Beschleuniger Intel i860 (33 MHz) zusätzlich zum Pixelstamp, für CAD/CAM/CAE usw.; *Modell 200 PXG Turbo* mit einer Dreifachkarte, die unter anderem als Grafikbeschleuniger den Prozessor i860 mit 40 MHz enthält. Mit letzterer Version soll die Verarbeitung von bis zu 400000 3D-Vektoren je Sekunde möglich sein. Damit eignet sie sich auch für Anwendungen im oberen Leistungsbereich wie Visualisierung und komplexe Animation.

Die Größe des Hauptspeichers liegt zwischen 8 und 120 MByte; der Systempufferspeicher hat 128 KByte. Eine Neuheit der „5000" ist der jetzt von DEC gebotene TURBOchannel als offene Desktop-Schnittstelle bzw. E/A-Kanal. Dieses neue Bussystem soll Datenübertragungsraten von 100 MByte/s erlauben. In der DECstation 5000 gibt es drei optionale Kanäle, von denen jeder eine Karte unterstützt. Diese können zur Steigerung der Grafikfähigkeiten oder zur Kopplung der Peripherie bzw. der Kommunikationsschnittstellen, beispielsweise VME, SCSI, FDDI, Ethernet, dienen.

Gleichzeitig mit der neuen Workstation wurde die neue Unix-Version von DEC, Ultrix 4.0, angekündigt, die Posix- und X/Open-Standards berücksichtigt. Als universelles Mehrbenutzersystem bzw. als Serversystem kommt die DECstation unter der Bezeichnung **DECsystem 5000, Modell 200** auf den Markt. Eine plattenlose DECstation soll mit 8 MByte Hauptspeicher, 19-Zoll-Monochrommonitor, Ethernet-Anschluß sowie Ultrix- und DECwindows-Lizenz für unter 30000 DM erhältlich sein.

Mitte Juni verknüpfte Hewlett-Packard seine eigene Workstationslinie mit der des 1989 übernommenen Herstellers Apollo Computer. Ein Ergebnis ist die neue **HP Apollo 9000 Serie 400**, die Modelle mit Leistungen zwischen 12 und 26 MIPS umfaßt und kompatibel zu den bisherigen HP- und Apollo-Workstations (mit Motorola-68000-Prozessor) ist.

Zum Einstieg ist das *Modell 400 dl* vorgesehen: ausgestattet mit Motorola 68030, 50 MHz und 12 MIPS Leistung bzw. 68040, 25 MHz und 20 MIPS. Die *Modelle 425t und 400t* leisten ebenfalls bis zu 20 MIPS bzw. 3,5 MFLOPS; sie verfügen über bis zu 64 MByte RAM, zwei integrierte 200-MByte-Harddisks, SCSI-, RS-232-C- und Centronics-Schnittstellen. Alle Versionen sind Auftischmodelle, im Gegensatz zu den im Beistellgehäuse untergebrachten *Modellen 433s und 400s*. Das Modell 433s ist insofern bemerkenswert, als es zwar mit dem 50-MHz-Motorola-68030 ausgestattet ist, aber dem EISA-Bus entspricht und damit kompatibel zum PC/AT- bzw. ISA-Bus sein soll. Die Leistung beträgt 12 MIPS; die des Modells 400s mit 33-MHz-68040-Prozessor 26 MIPS bzw. 4,5 MFLOPS.

Mit diesen Workstations kündigte HP auf Basis des 68040 auch eine neue Familie von Grafiksystemen, **HP Apollo VRX**, an. Deren Leistung soll bei bis zu einer Million 3D-Vektoren/s liegen.

Auch die Prozessorenhersteller selbst fertigen bekanntlich Computer, die mit ihren Prozessoren ausgestattet sind; und sei es auch nur als OEM-Modelle wie beispielsweise bei Intel. So kündigte die Firma bereits Anfang des Jahres als einsatzfertige Systeme auf der Basis der Prozessoren 386SX, 386DX und 486 die **MicroSystem Serie 1000** (386SX, 16 MHz, 3 MIPS, 4 MByte RAM, integrierte HD; als Einstiegsmodell), **Serie 3000** (386, 25 MHz, 6 MIPS, 64 KByte Cache, 8 MByte RAM, integrierte HD; als Mittelklasseangebot) und **Serie 4000** (486, 25 MHz, 12 MIPS, 8 KByte Cache on Chip, 8 MByte RAM, integrierte HD) an.

Der Entwickler der R3000-RISC-Prozessoren, die Firma MIPS Computer, hat ebenfalls unlängst eine neue Workstation-Serie sowie einen Server angekündigt. Die Workstation **Magnum 3000** basiert auf dem Prozessor R3000 und soll 3,6 MFLOPS bzw. 17,8 SPECmark leisten. Das Grundmodell hat 8 MByte RAM, einen 17-Zoll-Monochrom-Monitor, aber keine Plattenlaufwerke. OEM-Kunden können sich jedoch verschiedene Modelle nach Bedarf konfigurieren und aufrüsten.

Auch von anderen Firmen wie Siemens, Bull oder Unisys gibt es Produktankündigungen, auf die hier nicht mehr eingegangen werden kann – verfolgen Sie bitte dazu unsere Rubrik Entwicklungen und Tendenzen, in der wir regelmäßig besonders interessante Produkte vorstellen. *MP-We*

Massendaten mobil

Bereits seit längerem wird dem Ableger der Musik-Compact-Disk, der CD-ROM (Compact Disk-Read Only Memory) der große Durchbruch prophezeit. Dies ist einerseits verständlich, da auf der wechselbaren 5 1/4-Zoll-Scheibe immerhin über 500 MByte untergebracht werden können. Das entspricht etwa der Kapazität von 800 Disketten mit je 800 KByte. Oder 300 000 A4-Schreibmaschinenseiten. Natürlich lassen sich auch digitalisierte Bilder oder Musik auf der CD-ROM speichern. Andererseits resultiert aus der Tatsache, daß der Anwender die Platten nur lesen kann, also auf vorgegebene Informationen angewiesen ist, der bekannte Henne-und-Ei-Effekt: Computerhersteller und Anwender wollen die Laufwerke erst dann einsetzen, wenn genügend Applikationen auf CD-ROM verfügbar sind. Die CD-ROM-Anbieter wiederum zögern, so lange es für die mögliche und ökonomisch notwendige Massenproduktion der Scheiben auf dem Markt nicht genügend Laufwerke, sprich potentielle Kunden gibt. Dennoch sollen bereits an die 1000 CD-ROM-Titel lieferbar sein: vor allem Nachschlagewerke wie Telefonbücher, Patentinformationen, Bildarchive, Bestellkataloge, aber auch Servicekataloge von Herstellern. Und vielleicht gelingt es der CD-ROM, sich als nur lesbarer Speicher gegenüber den jetzt aufkommenden wiederbeschreibbaren optischen Medien ebenso zu behaupten wie die heute immer noch beliebte Schallplatte gegen die Konkurrenz des Magnettonbandes.

Vor diesem Hintergrund ist die Idee der kleinen Westberliner Firma Lotos Computer bemerkenswert, mit der zunehmenden Mobilität des Computers dem Anwender auch CD-ROM-Informationen zur Verfügung zu stellen – also einen Laptop mit CD-ROM-Laufwerk zu versehen. Aus der Idee wurde 1989 Realität in Form des **CD-ROM-Laptop;** ausgestattet mit 80286-Prozessor (12 MHz), 1 bis 4 MByte RAM, 40-MByte-Festplatte, 3,5-Zoll-/1,44-MByte-Floppy und einem 635-MByte-CD-ROM-Laufwerk. Erst zur diesjährigen CeBIT präsentierte ein weiterer Hersteller, die Firma Goupil, ebenfalls einen Laptop mit integriertem CD-ROM-Laufwerk; und zwar eine Version des bereits bekannten 80826-Golf. Da hatten die zwanzig Lotos-Leute jedoch schon eine Steigerung ihrer Idee – im Sinne von Miniaturisierung – parat: den **Infoman** (Bild 1), die Kopplung eines CD-ROM-Laufwerkes mit dem zur Zeit vielleicht kleinsten MS-DOS-Computer, dem Poqet PC. Dabei werden sowohl der Notizbuch-PC als auch das CD-ROM-Laufwerk auf einem von Lotos entwickelten und mit einem speziellen Adapter versehenen Grundgerät leicht wechselbar angeordnet. Das heißt, sowohl der Computer als auch das Laufwerk können wahlweise separat benutzt werden. Zusätzlich ist im Infoman eine 20- oder 40-MByte-Festplatte integriert.

Das Ganze wiegt dann einschließlich Akku knapp zwei Kilogramm und hat die Abmessungen von 28 x 23 x 4 cm^3. Der Computer selbst, eine Entwicklung der amerikanischen Firma Poqet Computer Corporation, wiegt knapp 450 g, hat eine Grundfläche von 22 x 11 cm^2 und eine Höhe von etwa 2,5 cm (Bild 2).

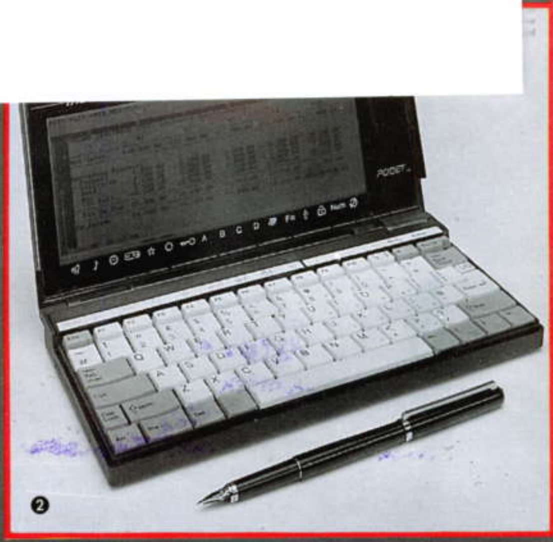
Der LC-Bildschirm ist aufklappbar und präsentiert die PC-üblichen 25 Zeilen mit je 80 Zeichen: die Auflösung beträgt bis zu 640 x 200 Punkten (CGA und MDA). Die Tastatur entspricht der erweiterten PC/AT-Tastatur, bei Schreibmaschinenausführung gibt es 77 Tasten. Herz des Poqet PC ist ein Prozessor Intel 80C88, der mit 2 bis 7 MHz getaktet werden kann. Serienmäßig ist der PC mit 512 KByte RAM und 640 KByte ROM ausgestattet (in dem unter anderem das Betriebssystem MS-DOS 3.3, GW-Basic, das BIOS und bereits sieben Anwendungs- und Systemprogramme – beispielsweise Textverarbeitung, Adressenverwaltung, Terminplaner oder Dateiverwaltungs-/Datenaustausch-Utilities – untergebracht sind). Wegen der geringen Abmessungen des Computers muß auf integrierte Disketten- oder Festplattenlaufwerke verzichtet werden. So werden der ROM mit Laufwerk C angesprochen, 20 KByte RAM standardmäßig als RAM-Disk, mit Laufwerk D. Zusätzliche Speicherkapazität erlangt man mittels Speicherkarten von Scheckkartengröße – als RAM gegenwärtig je 512 KByte, künftig 1 MByte, als ROM 2 MByte –, für die es zwei Aufnahmeschlitze im Gehäuse gibt und die als Laufwerk A und B angesprochen werden können. Eine RAM-Karte ist im Lieferumfang bereits enthalten. Darüber hinaus kann über den Connector, der als serielle, parallele oder Floppy-Schnittstelle dient, kommuniziert werden. Beispielsweise mit dem als externes Laufwerk lieferbaren Poqet Floppylaufwerk (3 1/2 Zoll, 1,44 MByte) oder einem anderen PC. Hervorzuheben ist die gegenüber anderen Notebook-PCs und Laptops sehr lange Nutzungsdauer von bis zu 100 Stunden je Batteriesatz (2x AA Mignon), die mit speziellen konstruktiven und softwaremäßigen Mitteln erreicht wurde. So enthalten die im Zuge der Entwicklung des Poqet angemeldeten 15 Patente allein 54 Einzelansprüche, die sich auf den Betrieb mit gewöhnlichen Mignon-Batterien beziehen.

Der Poqet PC (als Einzelgerät) soll Ende des Jahres für etwa 3950 DM (+ MwSt) erhältlich sein.

Bei dem zweiten Bestandteil des Infoman handelt es sich um ein CD-ROM-Laufwerk der Firma NEC mit einer Kapazität von 640 MByte, einer mittleren Zugriffszeit von 500 ms und einer Datentransferrate von 176,4 KByte/s. Auch das Laufwerk ist leicht auswechselbar, so daß es separat – auch für Audio-CDs – genutzt werden kann. Außerdem besteht damit die Möglichkeit, den Infoman anstelle des CD-ROM-Laufwerkes wahlweise mit einem wechselbaren Diskettenlaufwerk zu versehen, das zur Zeit bei Lotos in der Entwicklung ist. Bemerkenswert ist auch, daß in den Seriengeräten – deren Produktion im IV. Quartal in einem Betrieb in der DDR anlaufen soll – 128 KByte zusätzlicher RAM enthalten sind, womit die vom Poqet PC nutzbare RAM-Kapazität effektiv 640 KByte beträgt. Allerdings erhöht sich damit der – bisher unverbindlich genannte – Preis gegenüber früheren Angaben auf etwa 8200 DM.

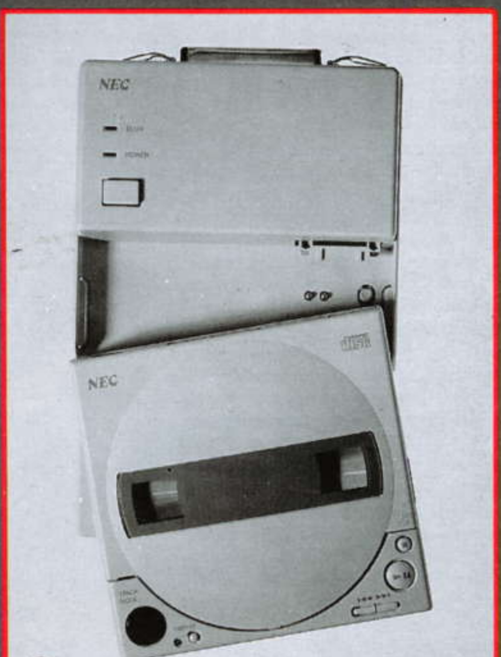
Eine weitere Variante, CD-Massendaten unterwegs oder an Rechnern nutzen zu können, die über kein eigenes Laufwerk verfügen, stellte zur CeBIT '90 die Firma NEC vor. Das **CDR-35** (Bild 3) besteht aus den Komponenten CD-ROM-Abspielgerät und Schnittstelleneinheit. Die Stromversorgung erfolgt über den 10,8-Volt-Akku- bzw. Batterieteil oder das Netzteil. Das Gerät wiegt (ohne Akku) 900 g und mißt 149 x 66 x 224 mm^3 (B x H x T). Es ist mit einer SCSI-Schnittstelle ausgestattet und kann mittels Adapter an PC/XTs, PC/ATs, Mikrokanal-PCs oder Apple Macintoshs angeschlossen werden. Anpassungen für verschiedene Laptops sind in Vorbereitung.

MP-We



❷

❶



❸

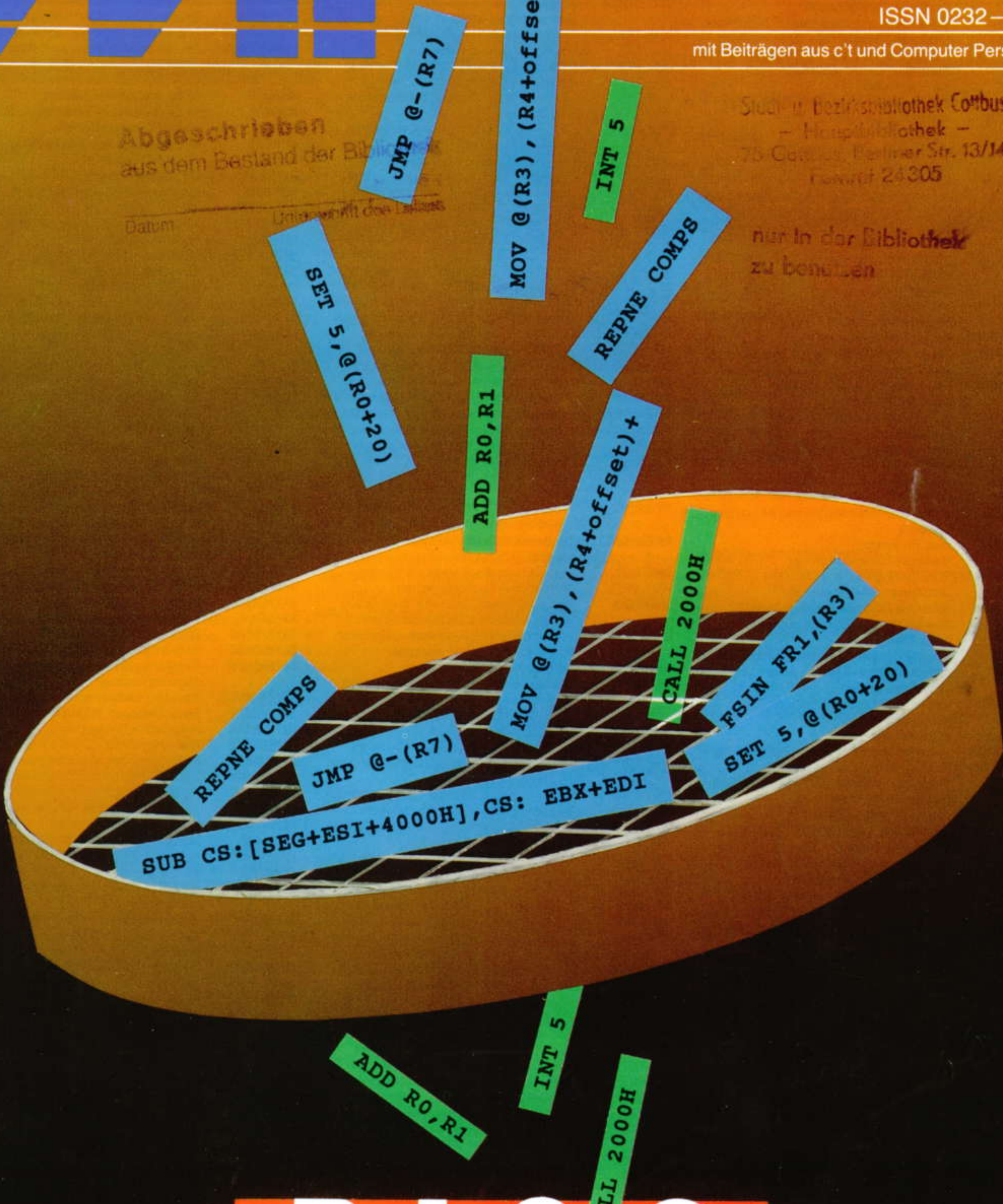
ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

DM 5,–
Heft 11 · 1990

MP Mikroprozessortechnik

Verlag Technik Berlin
ISSN 0232–2892

mit Beiträgen aus c't und Computer Persönlich



RISC

Weniger ist mehr

Referenzkarte
Nantucket Tools II

Anmerkung: Die Nantucket Tools II sind eine Sammlung von Funktionen für den Clipper-Compiler Sommer 87 und stehen auch für die Version 5.0 zur Verfügung. Ein Stern vor dem Namen einer Funktion zeigt an, daß sie in den Nantuk-ket Tools I noch nicht verfügbar war.

Fenster-Funktionen

Funktion	Beschreibung
WOPEN	eines von maximal 255 Fenstern öffnen
WSELECT	ein bestimmtes F. aktivieren
WCENTER	F. in sichtb. Bereich zurückholen
WMODE	Unter-Rand-Schiebe-Modus ein/aus
WBOX	Umrahmen des aktiven F.
WFORMAT	Ausgabeber. innerhalb eines F. bestimmen
WBOARD	nutzbaren Bildschirmber. für F. festlegen
WMOVE	F. an eine bestimmte Pos. bewegen
WCLOSE	das aktive Fenster schließen
WACLOSE	alle F. schließen
WSTEP	Schrittweite f. interaktive F.-bewegungen
WSETMOVE	Interaktiver Verschiebe-Modus erlaubt/gesperrt
WROW	oberste Zeilenposition des Fensters
WCOL	äußerste linke Spaltenpos. des F.
* WLASTROW	unterste Zeilenpos. des. F.
* WLASTCOL	äußerste rechte Spaltenpos. des F.

Der erweiterte Treiber NTOOLD87.OBJ

Funktion	Beschreibung
* CGA40	40-Zeichen-Modus (Farbe/Monochrom) ein
* CGA80	80-Zeichen-Modus (Farbe/Monochrom) ein
* EGA43	43-Zeilen-Modus (EGA) ein
* VGA50	50-Zeilen-Modus (VGA) ein
* MONOCHROME	Monochrom-Modus ein
* SETCRMODE	neuen Video-Modus als numerischen Wert einstellen
* GETSDRMODE	liefert aktiven Video-Modus als numerischen Wert
* GETMODE	liefer aktuellen Bildschirmmodus als Funktionsname
* MAXPAGE	höchste Seite (z. B. für SETPAGE(), PAGECOPY())
* GETPAGE	aktuelle Bildschirmseite abfragen
* SETPAGE	neue Bildschirmseite anwählen
* PAGECOPY	eine Bildschirmseite (page) in eine andere kopieren
* SETMAXCOL	max. Zeilenanz., die Clipper unterstützen soll
* SETMAXROW	max. Zeilenanz., die Clipper unterstützen soll
* GETPBIOS	Wird über DOS oder BIOS gedruckt?
* SETBIOS	über BIOS drucken; Time-out kann bestimmt werden
* GETPXLAT	aktuelle Drucker-Übersetzungstabelle holen
* SETPXLAT	individuelle Kode-Übersetzungen für Druckerausgaben
PRINTERROR	Kennziffer für Fehler bei letztem Ausdr.
GETCURSOR	Kursorform abfragen
SETCURSOR	Kursorform setzen
* GETBOXGROW	Wie werden die Fenster zur Zeit geöffnet
* SETBOXGROW	Fenster mit Exploding-Effekt öffnen
* GETTAB	Einstellg. d. Clipper-Tabulatoren ermitteln
* SETTAB	Bildschirm-Tabulatoren für Clipper-Ausgaben setzen
* SETBELL	Tonfrequenz und -dauer für CHR(7) oder Clipper-Beep setzen
DSETTYPE	Größe des gesetzten Tastaturpuffers
* DSETKBIOS	Tastaturabfrage über BIOS
* SETKKXLAT	Tastenkodes umdefinieren o. Tasten sperren
* GETKKXLAT	akt. Tastatur-Übersetzungstab. ermitteln
KEYSEND	Zeichen an Clipper-Tastatur-Puffer senden
KEYREAD	abgearbeitete Zeichen aus Puffer lesen
INPUTMODE	momentaner Eingabemodus
SETLINES	Zeilenzahl, nach d. bei *List* gewartet wird
* GETLINES	Den Wert von SETLINES abfragen
DSETNOLINE	nächsten Zeilenvorschub auf Bildsch. ignor.
DSETWINDOW	externe Funkt. u. Progr. in F. umlenken
DSETWINDEB	Debug auf ges. Bildsch. o. im akt. Fenster
DSETQFILE	Protokolldatei bei ordnungsgemäßen Programmende anlegen

Funktionen für die seriellen Schnittstellen

Funktion	Beschreibung
COM_OPEN	Schnittstellen öffnen, Puffer init.
COM_INIT	Schnittstellenparameter einstellen
COM_CLOSE	Puffer löschen, Schnittstelle schließen
COM_HARD	Hardware-Handshake (autom. CTS)
COM_SOFT	Software-Handshake (autom. XON/XOFF)
COM_SOFT_R	Ist ein XOFF eingegangen?
COM_SOFT_S	Wurde vom Puffer gesendet?
COM_COUNT	Anzahl empfangener Zeichen im Puffer
COM_READ	Zeichen aus dem Puffer lesen
COM_FLUSH	Empfangspuffer löschen
COM_SEND	Daten (auch int. gesteuert im Hintergr.) senden
COM_RING	Leitung abfragen (Ring)
COM_CTS	Clear To Send abfragen
COM_DSR	Data Set Ready abfragen
COM_DTR	Data Terminal Ready abfragen/setzen
COM_RTS	Request To Send abfragen/setzen
COM_MSR	Modem Status Register auslesen
COM_MCR	Modem Control Register auslesen/setzen
COM_LSR	Line Status Register auslesen/setzen
COM_BREAK	Break-Zustand auf Sendeleitung erzeugen
COM_KEY	Schnittstelle durch Keytraps überwachen
COM_EVENT	Welches Schnittstellen-Ereignis hat den Keytrap ausgelöst?
* COM_SETIO	Basisadresse eines Ports setzen
* COM_GETIO	Basisadresse eines Ports ermitteln
* COM_SETIRQ	Interrupt-Request eines Ports setzen
* COM_GETIRQ	Interrupt-Request eines Ports ermitteln
* COM_SFLUSH	Hintergrund-Sendepuffer löschen
* COM_SCOUNT	Zahl der noch im (Hintergrund)-Sendepuffer enthaltenen Zeichen
* COM_SKEY	Schnittstelle im Sendemodus durch Keytrap überwachen
* COM_REMOTE	Zeichen bestimmen, das den Sendepuffer extern löscht
* COM_SMODE	aktuellen Zustand des Hintergrund-Sendesystems ermitteln
* COM_ERRCHR	Zeichen bestimmen, das fehlerhaft empfangene Zeichen ersetzt
* COM_DOSCON	Bildschirmausgabe über DOS (Terminal-Emulation über ANSI.SYS)
* COM_NUM	Nr. der höchsten, verfügbaren seriellen Schnittstelle
* COM_CRC	Cyclic Redundancy Check (CRC) eines Strings berechnen
* XMOBLOCK	Block für X-Modem-Übertragung generieren (CRC o. Checksum)
* XMOCHECK	eingegangenen X-Modem-Block prüfen (CRC o. Checksum)

Zeichenkettenfunktionen

Funktion	Beschreibung
LTOC	logischen Wert in Zeichen umwandeln
ATNUM	n-tes Auftreten eines Strings im anderen
CSETATMUPA	Art d. Mehrfach-Durchgangs bei AT-Funkt.
BEFORATNUM	Stringanteil vor n-tem Auftreten einer Seq.
AFTERATNUM	Stringanteil hinter n-tem Auftreten einer Seq.
ATREPL	Sequenzen suchen und ersetzen
NUMAT	Anzahl des Auftretens einer Sequenz
NUMTOKEN	Zahl von Token in einem String
TOKEN	n-tes Token aus einem String holen
* TOKENSEP	Seperator vor o. hinter Token ermitteln
TOKENUPPER	ersten Buchst. eines Token in Großbuchst.
TOKENLOWER	ersten Buchst. eines Token in Kleinbuchst.
ATTOKEN	Auftreten d. n-ten Token in einem String
* TOKENINIT	Zeichenkette für TOKENNEXT() init.
* TOKENNEXT	liefert d. nächste mit TOKENNEXT ang. Token
* TOKENEND	Prüfen, ob bei TOKENNEXT() weitere Token verfügbar
RANGEREPL	Zeichen über Bereich v. ASCII-Kodes ersetzen
CHARREPL	best. Zeichen durch andere ersetzen
* WORDREPL	best. Doppel-Zeichen ersetzen
CHARRELA	Pos. zwei versch. Zeichen in zwei versch. Strings ermitteln u. bei Gleichheit anz.
CHARRELREP	alle mit CHARRELA ermittelten übereinst. Pos. mit best. Zeichen ersetzen
CHARREM	bestimmte Zeichen entfernen
CHARONLY	alle Zeichen bis auf bestimmte entfernen (Schnittmenge)
* WORDONLY	alle Doppelzeichen bis auf best. entfernen (Schnittmenge)
CHARONE	Mehrfachzeichen zu einem reduzieren
* WORDONE	Mehrfachdoppelzeichen zu einem Doppelzeichen reduzieren
CHARSORT	Zeichen/Seq. innerhalb d. Strings sortieren
CHARMIRR	Zeichen innerhalb d. Strings spiegeln
CHARLIST	Zeichenliste, jedes Zeichen in einem String nur noch einmal
* CHARMIX	zwei Strings miteinander vermischen
CHARODD	Zeichen d. unger. Pos. eines String
CHAREVEN	Zeichen d. geraden Pos. eines String
REMALL	Zeichen li. u. re. im String entfernen
REMLEFT	Zeichen li. im String entfernen
REMRIGHT	Zeichen rechts im String entfernen
JUSTLEFT	Zeichen von li. nach re. im String
JUSTRIGHT	Zeichen von re. nach li. im String
PADLEFT	String li. mit Zeichen auf best. Länge auffüllen
PADRIGHT	String re. mit Zeichen auf best. Länge auffüllen
COUNTLEFT	best. Zeichen am Stringanfang zählen
COUNTRIGHT	best. Zeichen am Stringende zählen
EXPAND	Leerzeichen in einen String einfügen
CENTER	String zentrieren
REPLLEFT	Zeichen li. im String ersetzen
REPLRIGHT	Zeichen re. im String ersetzen
REPLALL	Zeichen li. u. re. im String ersetzen
TABEXPAND	Tabulatoren in Leerzeichen umrechnen
* TABPACK	Leerzeichen in Tabulatoren umrechnen
NUMLINE	Zeilenzahl bei Ausgabe eines String
MAXLINE	größte Zeilenbreite bei Ausg. eines String
ASCIISUM	Summe der ASCII-Werte eines Strings
CHECKSUM	spez. Prüfsummen-Algorithmus für Strings
POSLOWER	Pos. des ersten Kleinbuchst.
POSUPPER	Pos. des ersten Großbuchst.
POSALPHA	Pos. des ersten Buchst.
POSRANGE	erstes Zeichen in einem Bereich von zwei ASCII-Kodes
POSREPL	Z. ab einer Pos. durch andere ersetzen
POSINS	Z. an einer Pos. in einem String einfügen
POSDEL	Z. an einer Pos. eines Strings löschen
POSDIFF	erste Pos., ab der zwei Strings sich unterscheiden
ASCPOS	ASCII-Kode eines Zeichens an best. Pos.
VALPOS	num. Wert eines Zeichens an best. Pos.
ADDASCII	zu jedem ASCII-Kode einen Wert addieren
CHARADD	ASCII-Kodes korrespondierender Zeichen zweier Strings addieren
CHARAND	ASCII-Kodes korrespondierender Zeichen zweier Strings über UND verknüpfen
CHAROR	ASCII-Kodes korrespondierender Zeichen zweier Strings über ODER verknüpfen
CHARXOR	ASCII-Kodes korrespondierender Zeichen zweier Strings über XOR verknüpfen
CHARNOT	ASCII-Wert eines Strings negieren (Komplementärwert)
STRSWAP	Austauschen zweier Strings
* CRYPT	Strings verschlüsseln/entschlüsseln
* WORDTOCHAR	Doppelzeichen in einfache wandeln
* CHARSWAP	benachbarte Zeichen vertauschen
* WORDSWAP	benachbarte Doppelzeichen vertauschen
* CHARPACK	Zeichenkette komprimieren (packen)
* CHARUNPACK	Zeichenkette dekomprimieren (entpacken)
* CHARSPREAD	Zeichenkette zu Blocksatz dehnen

Zahlen- und Bit-Manipulationen

Funktion	Beschreibung
NUMAND	16-Bit-AND zweier Zahlen
NUMXOR	16-Bit-XOR zweier Zahlen
NUMOR	16-Bit-OR zweier Zahlen
* NUMROL	16-Bit-Links-Rotation einer Zahl
NUMNOT	16-Bit-NOT einer Zahl
NUMHIGH	Die oberen 8 Bit einer 16-Bit-Zahl
NUMLOW	Die unteren 8 Bit einer 16-Bit-Zahl
NUMCOUNT	interner Zähler
INTNEG	16 Bit mit Vorzeichen-Bit-Interpret.
INTPOS	16 Bit als positiver Wert
ISBIT	Bits in einer Zahl testen
SETBIT	Bits in einer Zahl testen
INFINITY	Die größtmögliche Zahl (2^{1023})
EXPONENT	Exponent einer Gleitkommazahl (Basis 2)
MANTISSA	Mantisse einer Gleitkommazahl (Basis 2)
RANDOM	eine Zufallszahl generieren
* RAND	eine Zufallszahlenfolge generieren
* CELSIUS	Temp. v. Fahrenheit in Celsius umrechnen
* FAHRENHEIT	Temp. v. Celsius in Fahrenheit umrechnen
LTON	log. Wert in num. konvertieren
NTOC	num. Wert in String konvertieren bei best. Zahlenbasis
CTON	Zeichenkette in num. Wert konvertieren bei best. Zahlenbasis
* FTOC	Gleitkommazahl in 8-Zeichen-String umw.
* CTOF	8-Zeichen-String in Gleitkommazahl umw.
* CTOBIT	Zeichenfolge in Bitmaske umwandeln
* BITTOC	Bitpositionen in Zeichen konvertieren

Bildschirmfunktionen

Funktion	Beschreibung
STRSCREEN	String mit Attributen auf Bildsch. bringen
SCREENSTR	String mit Attributen vom Bildsch. lesen
SAYSCREEN	Bildschirmausg. ohne Attributänderung
* SAYDOWN	Bildschirmausg. senkrecht nach unten
* SAYSPREAD	Bildschirmausg. aus der Mitte heraus
* SAYMOVEIN	Bildschirmausg. hineinschieben (Effekt)
* CLEARSLOW	Fenster von außen nach innen mit Verzögerung löschen
* SCREENMARK	String auf Bildsch. suchen, mit Attrib. mark.
* SETSCRSTR	Bildschirmausg. in Zeichenkette umlenken
* GETSCRSTR	umgelenkte Bildschirmausg. übernehmen
SCREENFILE	Bildschirminhalt in Datei schreiben
FILESCREEN	Bildschirminhalt aus Datei lesen
SCREENMIX	Zeichen u. Attribute zu Bild mischen
COLORREPL	nur best. Bildschirmattribute ändern
COLORWIN	nur best. Attribute im Fenster ändern
* INVERTWIN	Attribute im Fenster invert.
* CHARWIN	nur best. Zeichen im Fenster ändern
* UNTEXTWIN	Bereich v. Zeichen im Fenster tauschen
CLEAREOL	bis Zeilenende mit best. Zeichen löschen
CLEOL	bis Zeilenende mit Standardzeichen u. -attrib. löschen
CLEARWIN	Bereich mit best. Zeichen löschen
CLWIN	Bereich mit Standardzeichen u. -attribut löschen
ISHERCULES	Hercules-Karte vorh. bzw. emulierbar?
ISEGA	EGA-Karte vorh. bzw. emulierbar?
* ISCGA	CGA-Karte vorh. bzw. emulierbar?
* ISMCGA	MCGA-Karte vorh. bzw. emulierbar?
* ISPGA	PGA-Karte vorh. bzw. emulierbar?
* ISMONO	Monochrome-Karte vorh. bzw. emulierbar?
* ISVGA	VGA-Karte vorhanden?
VIDEOSETUP	Bildschirmmodus bei Systemstart
* VIDEOTYPE	Bildschirmkarten-Typ
* VIDEOINIT	Video-Syst. nach Progr.-aufruf mit RUN neu init.
NUMCOL	Zahl der Bildschirm-Spalten
MAXCOL	größte unter Clipper verfügb. Spaltenzahl
MAXROW	größte unter Clipper verfügb. Zeilenzahl
SCREENSIZE	Größe des verfügbaren Bildschirms
SCREENATTR	Attribut auf Pos. Zeile, Spalte ermitteln
INVERTATTR	Vorder- u. Hintergr.-Attribut vertauschen
SETCLEARA	Vorgabeattribut für Bildsch.-löschen setzen
GETCLEARB	Vorgabezeichen für Bildsch.-löschen abfr.
SETRC	Zeile u. Spalte für Clipper-Kursor setzen
* SAVECURSOR	Kursorposition u. -form sichern
* RESTCURSOR	Kursorposition u. -form wiederherstellen
NTOCOLOR	num. Wert in Farbwert konvert.
COLORTON	Farbwert in num. Wert konv.
* CHARPIX	Zahl d. Pixel-Zeilen pro Zeichen
* MAXFONT	Anz. der verfügbaren Fontbereiche
* GETFONT	akt. Font in String kopieren
* SETFONT	Font aus String heraus verändern
* FONTLOAD	Fonts für EGA/VGA aus Datei laden
* FONTSELECT	Font-Tab. für nomale/intens. Ausg. best.

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
4. Jahrgang · ISSN 0232–2892

Mikroprozessortechnik, Heft 11 · 90

Verlag Verlag Technik GmbH, Oranienburger Str. 13/14, O – 1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd, Telefax: 2 87 02 59

Geschäftsführer Klaus Hieronimus

Verlagsleiter Zeitschriften Günter Fitzner

Redaktion Hans Weiß, Chefredakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2 87 02 03), Hans-Joachim Hill (Tel. 2870209); Sekretariat: Hannelore Kulik (Tel. 2870381)

Gestaltung Christina Bauer

Titel Frank-Norbert Beyer

Beirat Prof. Dr. sc. Thomas Horn, Prof. Dr. Bernd Junghans, Dr. Dietmar Keller, Prof. Dr. sc. Bernd-Georg Münzer, Prof. Dr. sc. Peter Neubert, Prof. Dr. sc. Dr. Michael Roth (Vorsitzender), Dr. Gerhard Schulze, Prof. Dr. sc. Manfred Seifart, Dr. Dieter Simon, Prof. Dr. sc. Dr. Jürgen Zaremba

Gesamtherstellung Märkische Verlags- und Druck-Gesellschaft mbH Potsdam

Anzeigenannahme Verlag Technik GmbH, Anzeigenabteilung, Oranienburger Straße 13/14, O–1020 Berlin; Tel. 2 87 02 54, 2 87 03 09, Telefax: 2 87 02 54

Anzeigenpreise Preisliste Nr. 1 vom 1. Juli 1990

Erfüllungsort und Gerichtsstand Berlin-Mitte. Der Verlag behält sich alle Rechte an den von ihm veröffentlichten Aufsätzen und Abbildungen, auch das der Übersetzung in fremde Sprachen, vor. Auszüge, Referate und Besprechungen sind nur mit voller Quellenangabe zulässig.

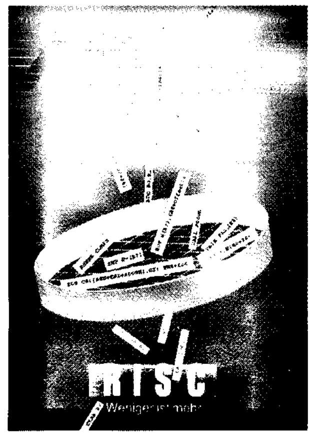
Redaktionsschluß 12. September 1990

AN (EDV) 49837

Erscheinungsweise monatlich

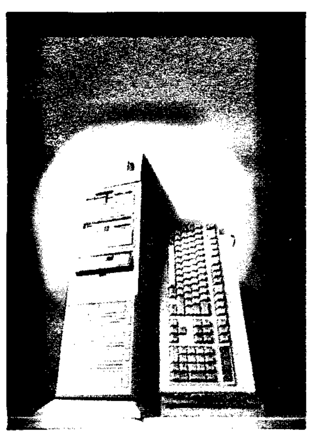
Heftpreis 5,– DM, Abonnementspreis vierteljährlich 15,– DM; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
Verlag Technik, Abteilung Absatz, PSF 201, O–1020 Berlin und sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propagandit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSFR:* PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *Republik Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Republik Ungarn:* P.K.H.I., Külföldi Előfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nôi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Buchhandel oder Verlag Technik GmbH



Zum Titelbild:

RISC ist die Abkürzung für Reduced Instruction Set Computer und bedeutet die Reduktion des Befehlssatzes und der Befehlsformate. Während bis in die 70er Jahre Computer mit immer komplexeren und größeren Befehlssätzen ausgestattet wurden, erkannte man etwa zu dieser Zeit, daß die Rechner rund 90 Prozent dieser Befehle nur selten nutzten. Die Konsequenz lautete: Auslese und führte zu RISC-Prozessoren, die mit weniger und kürzeren Befehlen wesentlich schneller arbeiten können. Besonders im Zusammenhang mit leistungsstarken Workstations werden sie heute oft genannt. Mehr dazu in unserem Beitrag auf der **Seite 4**.



Im Test:

Auf der **Seite 8** stellen wir Ihnen am Beispiel eines für uns getesteten LGI 486 die Klasse der PCs mit dem Prozessor 80486 vor. Wir wollen Ihnen damit helfen, die einzelnen Leistungsmerkmale denen anderer Rechnerklassen gegenüberzustellen, um sich nicht zu „verkaufen".

Nachdem in der jüngsten Vergangenheit mehrere Grenzen gefallen sind, nun von uns ein Beitrag zu diesem Thema aus der Sicht des PC-Nutzers. Die magische Grenze des nutzbaren RAMs unter MS-DOS von 640 KByte wurde schnell zu einer starken Einschränkung, insbesondere bei umfangreichen Anwendungsprogrammen oder bei der Nutzung vieler speicherresidenter Programme. Wie Sie endlich auch diese Grenze überwinden können, zeigt unser Beitrag auf der **Seite 25**.

Inhalt

Referenzkarte Nantucket Tools II — **2. US**

MP-Info — **2**

Rene Steiner:
Merkmale der RISC-Architektur — **4**

kennengelernt
Microsoft C 6.0 PDS — **7**

Uwe Schulze, Christian Hanisch:
Darf's auch ein 486er sein? — **8**

Bernd Matzke:
Mit der C-Shell doppelseitig drucken — **10**

Konrad Schultz:
Grundlagen des logischen Programmierens (Teil 2) — **11**

Andreas Stiller, Carsten Meyer:
Connections (Teil 1) — **14**

Wegbereiter der Informatik
Johann (John) Ludwig v. Neumann — **16**

MP-Kurs: — **17**
Ulrich Kriegel:
Lisp (Teil 3)

Uwe Holz:
Die Nantucket Tools II — **23**

Franz Otto:
Schlecht beraten — **24**

Hans-Georg Sanner:
Die Überwindung der RAM-Grenzen unter DOS — **25**

MP-Computer-Club — **30**
Olaf Matthäi:
Realtime-Clock-Modul für 8-Bit-Rechner
Roland Radandt:
Inhaltsverzeichnisse vergleichen
Hans Heukenkamp:
Schnelle Winkelfunktionen in Turbo-Pascal

MP-Bericht — **32**
Leipziger Herbstmesse 1990
Atari-Messe 1990

Entwicklungen und Tendenzen — **36**

kennengelernt
Volkswriter — **4. US**

Vorschau

Für das Heft 12/1990 bereiten wir für Sie unter anderem Beiträge zu folgenden Themen vor:

- Leiterplattenentwurf auf dem PC
- Speicherprogrammierbare Steuerungen
- kennengelernt: WordPerfect 5.1
- Quick-Pascal

Siemens-Schulungen

Die Siemens-Schule für Daten- und Informationstechnik Berlin bietet jetzt ihr Aus- und Weiterbildungsprogramm auch für Ex-DDR-Interessenten an. Neben dem gewohnten Schulungsangebot finden Einführungskurse in die allgemeine Betriebswirtschaftslehre auch am Informatik-Center Thüringen in Gera statt. Mit dem Arbeitsamt Berlin wurden Lehrgänge für unterschiedliche DV-Ausbildungsberufe und Umschulungen entwickelt, die einem Aufbaustudium gleichen. Ein Weiterbildungsprogramm mit der Kammer der Technik rundet das Spektrum ab.
Das Angebot umfaßt neun DV-Schwerpunkte und ermöglicht durch den modularen Kursaufbau eine individuelle Weiterentwicklung für jeden Interessenten. In Zusammenarbeit mit dem Arbeitsamt Berlin und der Siemens-Schule für Datentechnik sind Lehrgänge entwickelt worden, die Hochschulabgänger zu EDV-Spezialisten ausbilden. Folgende Ausbildungsberufe werden angeboten: Kommunikations-Organisator/-Programmierer, Informationsorganisator und Softwareentwickler.
Zugangsvoraussetzungen sind ein abgeschlossenes Hochschulstudium und das Bestehen des DV-Tests. Die Gebühren liegen bei einer Lehrgangsdauer von 12 bis 20 Monaten zwischen 18000 und 32000 DM. Eine Förderung durch das Arbeitsamt ist möglich. Die DI-Schule eröffnet Anfang '91 ihre Außenstandorte in Dresden und Leipzig. Weitere Schulungsangebote sind für Thüringen, Cottbus, Fränkfurt/Oder und Rostock geplant.
Interessenten wenden sich bitte an: Siemens AG, Schule für Daten- und Informationstechnik, Rohrdamm 85, W-1000 Berlin 13; Tel. (030) 386-4755 oder Zentralstelle Information, Pressestelle Berlin, Nonnendammallee 101, W-1000 Berlin 13; Tel. (030) 386-7121

Bestückungs-Service in 48 Stunden

Die FELA Multielectronic GmbH hat einen *Eildienst für Bestückung* eingerichtet. Alle Arten von Leiterplatten, konventionelle, SMD-, flexible und starrflexible Leiterplatten als Kleinserien und Prototypen, werden in 48 Stunden bestückt. Der Test erfolgt nach Absprache. Für Großserien beträgt die Lieferfrist zur Zeit 15 Arbeitstage für das erste Lieferlos.

Zusammen mit einem ebenfalls angebotenen Eildienst für Leiterplatten, ermöglicht der neue Bestückungs-Service die Fertigung von kompletten, einbaufähigen elektronischen Baugruppen je nach Leiterplattentyp zwischen zehn und fünfzehn Tagen. Beide Eildienste können auch unabhängig voneinander genutzt werden.
Eilige Konstruktionsarbeiten werden damit nicht mehr durch tage- oder gar wochenlanges Warten unterbrochen. Für den Eildienst werden die üblichen Unterlagen, wie Bauteilliste und Bestückungsplan benötigt. Bestückt wird auf denselben Anlagen wie bei Großserien. Die Muster haben Serienqualität.

Die Äpfel werden billiger

Die Firma Apple hat jetzt „im Zuge einer europäischen Preisanpassung" für verschiedene Macintosh-PCs deutliche Preisreduzierungen vorgenommen, und zwar beim SE/30 (2/40) von 8904 DM auf 6695 DM, beim SE/30 (4/40) von 9895 DM auf 7439 DM, beim IIci (4/40) von 15877 DM auf 12702 DM und beim IIci (4/80) von 16886 DM auf 13895 DM (jeweils plus MwSt., unverbindliche Preisempfehlung).
Zusätzliche Preisvergünstigungen gibt es nun auch für Studenten, neben Universitäten, Schulen und Lehrkräften, durch Nachweis mittels Immatrikulationsbescheinigung, beim Kauf des SE/30 (2/40) mit 5724 DM und des Personal LaserWriter SC mit 3263 DM (jeweils unverbindl. Preisempfehlung inkl. MwSt.).

Informationsquelle im Herzen Berlins

Vielleicht ist es auch Ihnen schon passiert, daß Sie vom Verlag Technik angekündigte Bücher oder auch die neuesten Ausgaben der Fachzeitschriften nicht in der Buchhandlung oder am Kiosk erhalten konnten. Zumindest denen, die in oder um Berlin wohnen oder zu einem – sehr empfehlenswerten – Abstecher hier weilen, kann jetzt geholfen werden: Der Verlag hat einen **Technik-Buch-Shop** eröffnet.
Unmittelbar neben dem Verlagsgebäude gelegen (Oranienburger Straße 12; Tel. 2870295), werden montags bis freitags von 9 bis 18 Uhr Bücher und Zeitschriften unseres Verlages angeboten. Und es gibt Informationen, ganz direkt und umfassend!

Informationsveranstaltung für Entwickler

Die VDI-Gesellschaft Entwicklung Konstruktion Vertrieb (VDI-EKV) – mit über 16000 zugeordneten VDI-Mitgliedern die größte Fachgesellschaft des Vereins Deutscher Ingenieure – bereitet eine Informationsveranstaltung vor, die im Februar 1991 im Prduktionstechnischen Zentrum Berlin durchgeführt wird. Angesprochen sind Hochschul- und Fachhochschullehrer sowie Ingenieure aus der Ex-DDR, die auf den Gebieten Entwicklung, Konstruktion, Getriebetechnik, Schwingungstechnik, technischer Vertrieb, Informationstechnik sowie Bürokommunikation tätig sind.
Ziel der Veranstaltung ist es, über die Aktivitäten der VDI-EKV – Erarbeitung von VDI-Richtlinien und VDI-Fachbüchern, Vorbereitung und Durchführung von Tagungen und Kongressen, Entwicklung von Seminaren – zu informieren und zur Mitarbeit in den entsprechenden Ausschüssen zu motivieren. Darüber hinaus soll die Bildung lokaler Arbeitskreise angeregt werden, mit denen Ingenieure in ihren Heimatorten in der Ex-DDR angesprochen und fachlich betreut werden können. Die ganztägige Veranstaltung ist kostenlos. Informationsmaterial und Bewirtung inklusive.
Interessenten werden gebeten, sich schriftlich oder telefonisch zu melden bei der Geschäftsstelle der VDI-Gesellschaft Entwicklung Konstruktion Vertrieb (VDI-EKV), Postfach 1139, W-4000 Düsseldorf 1; Tel. 0211/6214-240.

AST Research Deutschland mit Robotron im Geschäft

Mit zwei Gesellschaften der Unternehmensgruppe Robotron haben AST Research Deutschland und NBI Deutschland/BTO Text- und Datensysteme auf Initiative der Deutschen Bank ein Vertriebsabkommen unterzeichnet: Zum einen mit der CVU (Computer Vertriebs-Union, vormals Robotron-Vertrieb Berlin), und zum anderen mit der Robotron Anlagenbau GmbH in Leipzig. Beide Unternehmen sind seit Juli offizieller Lieferant für alle AST-Produkte von der Erweiterungskarte bis zum Personalcomputer für professionelle Anwendungen. Damit ist ein flächendeckender Vertrieb und Service für AST-Produkte gewährleistet. Die beiden Robotron-Unternehmen zählen derzeit 5700 bzw. 2700 Mitarbeiter und werden gezielt den Groß- und Endkundenbereich angehen. Den notwendigen Know-How-Transfer übernimmt die NBI/BTO einschließlich Produkttraining und der Belieferung mit schlüsselfertiger Anwendersoftware. NBI/BTO ist der regionale Vertriebspartner von AST in Ratingen und spezialisiert auf schlüsselfertige Systeme im Bereich der Bürokommunikation.

Neues von Acer

Die Acer-Gruppe und Altos Computer Systems gaben gemeinsam die Unterzeichnung ihres Zusammenschlusses bekannt. Altos, als weltweiter Marktführer im Bereich der vernetzten Mehrplatzsysteme angesehen, wird hierdurch Mitglied der Acer-Gruppe. Diese ist eine internationale Unternehmensgruppe, die im Industriesektor Mikrocomputer tätig und Taiwans größter PC-Produzent ist. Ihre Produkte werden in 70 Ländern von mehr als 10000 Fachhändlern vertrieben. Die in Deutschland tätige Acer-Cetec Computer GmbH (bis vor kurzem CE-TEC data Technology GmbH) eröffnete Ende April ihr erstes Vertriebsbüro in Ost-Berlin. Von hier aus werden die etwa 40 Acer-Vertragshändler in der Ex-DDR betreut.

Beratungstage für Softwareentwickler

Die Firmen GID (Gesellschaft für Informationstechnologie und Datenverarbeitung) Freiberg, ExperTeam Köln und Micro Focus München veranstalteten zwei Informationstage zum Thema „Initiative Zukunft – Professionelle Software-Anwendungsentwicklung mit COBOL ANSI 85" im Robotron-Schulungszentrum in Leipzig. Im Mittelpunkt der Plenumsveranstaltung stand die Frage, wie sich die Software-Produktion im Osten Deutschlands am schnellsten und effizientesten den neuen Anforderungen, die sich durch die politischen Veränderungen der letzten Monate ergeben haben, anpassen läßt. Der besondere Schwerpunkt von ExperTeam im Rahmen dieser Veranstaltung war der methodische Aspekt der Software-Entwicklung. Die Bedeutung der Strukturierten Programmierung für die Erstellung und vor allem die spätere Wartung von Softwaresystemen wurde anhand der Methode der Jackson-Strukturierten-Programmierung (JSP) besonders herausgestellt. Die Möglichkeit, Programme graphisch anhand von Baumdiagrammen zu entwickeln, wird heute durch PC-basierte Werkzeuge eindrucksvoll unterstützt. Um den Teilnehmern dies „live" zu demonstrieren, präsentierte ExperTeam im Rahmen der COBOL-orientierten Veranstaltung die JSP-Workbench, eine integrierte COBOL-Entwicklungsumgebung auf dem PC, die die Micro Focus COBOL/2-Workbench als Basis hat. Durch die Verlagerung der Entwicklungsarbeit auf den PC kann der Programmierer sehr viel komfortabler bei seiner Arbeit unterstützt werden. Mit der JSP-Workbench werden Programme nicht kodiert, sondern gra-

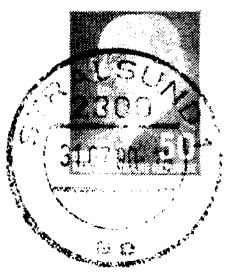
fisch entworfen. Die gesamte Programmlogik wird anhand von Jackson-Baumdiagrammen beschrieben. Aus diesen Programmentwürfen läßt sich automatisch der COBOL-Code und die Dokumentation generieren. Die Testunterstützung innerhalb der JSP-Workbench durch symbolisches Debugging auf Entwurfsebene soll in dieser Form bislang einmalig sein. Sie erlaubt es, vom ersten Entwurf eines Programms bis zum Abschluß der Testphase durchgängig mit einer anschaulichen grafischen Darstellungstechnik zu arbeiten.
Das Interesse der Teilnehmer an den Produktpräsentationen der JSP-Workbench und der Micro Focus COBOL/2-Workbench war außerordentlich groß. Vor allem stieß das Konzept der ganzheitlichen methodischen Programmentwicklung auf für westliche Verhältnisse unerwartet große Akzeptanz. Nach Meinung der Aussteller haben die „Ost"-Softwareentwickler einen entscheidenden Vorteil gegenüber den Entwicklern in der Bundesrepublik: Sie können mit modernster Technologie, mit anerkannten Methoden, mit einer vollstrukturierten und normierten Programmiersprache, unbelastet von „Programmruinen" der letzten Jahrzehnte, die es zu sanieren gilt, den Aufbau ihrer Systeme beginnen. In vielen Fällen wird es dabei möglich sein, das Zeitalter des Großrechners als Entwicklungsrechner zu überspringen und von Anfang an den Komfort eines PCs als Entwickler-Workstation zu nutzen.

Legalisierungs-Boom

Immer mehr westliche Firmen nutzen die Möglichkeit, ihre Anwendergemeinde in der Ex-DDR durch nachträgliches Lizensieren von „adaptierter Software" (oder wie immer man diese gern nannte) zu vergrößern. Über Borland (s. MP 4/90), Wordstar (s. MP 6/90), Jolo Data (s. MP 6/90), Ziegler Instruments (s. MP 7/90) und TommySoftware (s. MP 8/90) hatten wir schon berichtet. Auch Informix gehört mt Updates auf Smartware gegen Insys dazu.
In einer Aktion der Digital Equipment GmbH, München, erhalten Anwender von VAX-Rechnern und entsprechenden Nachbauten jetzt für 500 Mark pro Zentraleinheit eine Lizenz für die von ihnen bereits genutzte bzw. neueste Version des Betriebssystems VMS, des FORTRAN- und COBOL-Compilers sowie des Maskengenerators DECforms und der Datenabfrage- und -bearbeitungssprache VAX DATATRIEVE. Ebenfalls für 500 Mark bekommen Betreiber von PDP-11- und kompatiblen Rechnern eine Lizenz für die bereits genutzte bzw. die neueste Version des Betriebssystems RSX-11M-PLUS, der Compiler FORTRAN 77 und COBOL-81, des Maskengenerators FMS und der Abfrage-/Bearbeitungssprache DATATRIEVE. Für Original-Digital-Produkte können Serviceverträge mit dem Digital-Kundendienst geschlossen werden; für den Service von kompatiblen Rechnern verweist Digital auf die beiden Unternehmen Computer-Vertriebs-Union Berlin (CVU) und Integrierte Informations Systeme Dresden (I^2S).
Nachdem von Microsoft zunächst andere Töne kamen, treibt die Firma jetzt auch auf der Legalisierungswelle. Sie bietet in einer „einmaligen und kostengünstigen Aktion" Ex-DDR-Betrieben, die vor dem 1. Juli 1990 DOS-kompatible PCs erworben haben, an, bei Vorlage von Kopien der Kauf- oder Investitionsbelege elf Programme bei autorisierten Fachhändlern umzutauschen.
Auch Ashton Tate geht den Weg, Klones seiner Produkte nicht direkt, sondern über eine Umtauschaktion zu legalisieren. Bis Ende des Jahres können Besitzer von Redabas oder Ariadne diese Programme bei einem der über 70 Fachhändler gegen dBase III Plus bzw. Framwork eintauschen. Voraussetzung ist die Abgabe der Robotron-Originaldisketten samt Handbüchern. Die Umtauschpreise betragen jeweils 399,–DM (incl. MwSt.). Nach dem Jahreswechsel, so die Presseinformation, „wird Ashton Tate von seinem Copyright Gebrauch machen und alle Anwender in der DDR rechtlich belangen, die nicht mit einer lizensierten Produkt-Kopie arbeiten." Anwender, die eine „Produkt-Kopie" für nicht wenig Geld nach damals geltendem DDR-Recht erwerben mußten, werden dafür wenig Verständnis haben; zumal nichts darüber verlautbart wurde, daß die Firma ihre Interessen auch gegenüber dem Verursacher – den Rechtsnachfolgern des Robotron-Projekt Dresden – durchsetzen will. In puncto Public Relation wohl kaum eine kluge Entscheidung. Wie ernst es Ashton Tate mit den harten Bandagen gegenüber den Anwendern wirklich ist und ob damit überhaupt Aussicht auf einen Sieg besteht, darüber wollen wir Sie in einem der nächsten Hefte informieren.

Herzlichen Glückwunsch!

Das war sie also, unsere Leserumfrage 1990! Und das sind die Gewinner, die wir anhand der eingesandten Rückscheine aus Heft 7/90 ermittelten:
Der Hauptpreis, ein LGI 286 (ein PC der AT Klasse von der Firma Lucky Goldstar International), ging an Herrn *Christian Schulz* aus Leipzig.
Die Gewinner von je einem Textverarbeitungsprogramm Volkswriter der gleichnamigen Firma sind: H. Koschitzki, H.-H. Kruse, H. Fröhlich, S. Drockur, Th. Brix, U. Grabsch, A. Hartbrich, S. Engelschaft, Ch. Schulz, M. Bärtig, D. Futász, D. Ebenroth, K.-H. Manthey, O. Großmann, U. Knietzsch, C. Mündel, Th. Blaschczok, J. Petermann, H. J. Schmidt und M. Kaul.
Darüber hinaus verlosten wir 20 Lernprogramme, ebenfalls von Volkswriter, sowie je 25 Bücher *Wissensspeicher Mikrorechnerprogrammierung* und *Programmieren mit C* aus dem Verlag Technik. Die Preise wurden den Gewinnern inzwischen zugestellt.
Zum Verlosungsmodus: Eine Vielzahl von Lesern hatte uns auf die unglückliche Anordnung der Antwortscheine (Rückseite Messebericht) aufmerksam gemacht. Das kam für uns nicht unerwartet. Vor die Wahl gestellt, noch einen Monat länger zu warten oder die nach Redaktionsschluß nur auf dieser Seite noch mögliche Chance zu nutzen, entschieden wir uns angesichts der Entwicklung in der Medienlandschaft jedoch für die schnellere Lösung. Für die so vielen dadurch ungünstig zerschnittenen Hefte bitten wir unsere Leser hiermit aufrichtig um Entschuldigung. Es soll nicht wieder vorkommen!
Etliche Leser, die ihr Heft nicht zerschneiden wollten, sondern uns Kopien oder Abschriften schickten, befürchteten nun, nicht berücksichtigt zu werden. Dem ist nicht so. Allerdings schied wegen der unterschiedlichen Beschaffenheit und Größe der Zuschriften die zunächst geplante Variante aus, die Gewinnscheine aus dem bis oben hin gefüllten Verpakkungskarton des Computers direkt zu ziehen. So ließen wir den Hauptpreis selbst sein eigenes und das Schicksal der anderen Preise bestimmen: Per Zufallsgenerator-Programm und durch Tastendruck unserer Redaktinsmitarbeiterin als gute Fee wurden die laufenden Nummern der Gewinner-Einsendungen ermittelt. Objektiver und einer Computerzeitschrift angemessener kann es wohl kaum zugehen.
Rein subjektiv dagegen bestimmten wir 5 Empfänger von je einem „Wende-Buch" (ein empfehlenswertes Karikaturenbändchen zum Thema Wende aus dem Verlag Technik). Honorieren wollten wird damit wenigstens einige der vielen originellen Einsendungen. Da wäre zunächst der Brief von Herrn *Rathke* aus Stralsund, versehen mit einer Walter-Ulbricht-50-Pfennig-Marke. Vielen Dank für diese philatelistische Kostbarkeit.



Herr *Lotter* aus Suhl beendete seinen ausführlichen, mit konstruktiver Kritik versehenen Brief mit: *... möchte ich jetzt ausrufen: Wachse, meine MP, werde dick, preisgünstig, pünktlich und interessant!* Wir nehmen es als Auftrag an. Herr *Bialas* vermerkte neben dem Feld „privater Computeranwender" ein Fragezeichen und *Wenn ich gewinne.* Tut uns leid. Herr *Knecht* aus Bergholz-Rehbrücke schrieb uns: *Nun ja, für einen Monat bieten manch andere Magazine entschieden mehr fürs Geld ... natürlich auch mehr Schnickschnack, das stimmt schon, es bleibt aber trotzdem immer noch so viel Monat übrig, wenn die MP ausgelesen ist. Schade.* Finden wir auch. Als Überbrückung, bis bestimmt auch die MP dicker wird, immer mal ins Wendebuch blicken! Das letzte Büchlein ging an Herrn *Schröder* in Kahla, der seine Hinweise mit den prosaischen Worten abschloß: *Bleibt so und danke für die bisherige Unterstützung. Ihr wart das Licht im Dunkeln!*

Liebe Leserinnen und Leser! Wenn Sie die letzten Sätze gelesen haben, wird so mancher von Ihnen vielleicht denken, wir nutzten die Auswertung der Umfrage als billiges Eigenlob. In der Tat gab es unter den Einsendungen viele, die mit Kritik nicht sparten. Fast allen war – verständlicherweise – der Umfang der MP zu gering, manchem zu viel Theorie, manchem zu viel Bastelei im Heft, manch einer fand zu viele Themen zu fachspezifisch abgehandelt, und manch einem waren die meisten Beiträge zu allgemein. (Das breite Spektrum der MP ist aber vielleicht auch eine gute Übung zur Toleranz.) Die Zuschriften, in denen Forderungen ultimativ mit der Konsequenz der Abbestellung verbunden wurden, lassen sich jedoch an den Fingern einer Hand abzählen. (Natürlich bedenken wir auch die Meinungen der Nicht-Schreiber und deren Verhalten in bezug auf die Auflagenentwicklung.) Um ein Vielfaches häufiger waren dagegen die Briefe, in deren Zeilen sich die ehrliche Sorge um die Existenz der Zeitschrift und der Wunsch nach deren Erhalt widerspiegelt. Sätze wie: *Bleibt stark! Macht weiter so und haltet Euch tapfer! Laßt Euch von westdeutschen PC-Zeitschriften nicht unterkriegen!!! Viel Glück in der Marktwirtschaft! Beste Überlebenswünsche und im großen und ganzen so bleiben!* sprechen wohl eine deutliche Sprache. Den dutzendfach geäußerten Wunsch „Weiter so!" wollen wir so verstehen, daß wir die bisherige Herstellerunabhängigkeit und die Ehrlichkeit gegenüber unseren Lesern beibehalten. Bezogen auf das Heft und die redaktionelle Arbeit soll (und muß) sich natürlich durch die neuen Marktverhältnisse doch etliches ändern. Und diesbezüglich sind uns Ihre Hinweise eine große Hilfe. Günstig wirkte sich aus, daß wir Sie nicht nur Kreuzchen machen ließen, sondern die Gelegenheit zu verbalen Äußerungen einräumten. Obwohl dies mit erheblichem Auswertungsmehraufwand verbunden war. Schließlich wollten 1382 Zuschriften erst einmal gründlich gelesen und dann zur statistischen Auswertung – die noch läuft – erfaßt werden. Und all das neben der üblichen redaktionellen Arbeit. Wir möchten daher all diejenigen um Verständnis bitten, die in ihren Zuschriften (teilweise in längeren Briefen) Fragen oder Wünsche äußerten, wenn die Beantwortung noch etwas auf sich warten läßt. Wir haben uns über jede Zuschrift gefreut und bedanken uns auf diesem Wege bei jedem einzelnen für die Mitarbeit.
Auf die Auswertung der Umfrage hier im Detail einzugehen wäre noch zu früh. Einige der Hinweise, die die redaktionelle Arbeit betreffen, werden wir umgehend berücksichtigen; andere – insbesondere wenn sie mit materiellen oder personellen Aufwendungen verbunden sind – können sicher nur schrittweise verwirklicht werden. Schließlich ist auch eine Redaktion wie eine betriebliche Abteilung nur *ein* Teil eines VEB – Verzeihung: einer GmbH –, und marktwirtschaftliche Erkenntnisprozesse führen, wie Sie vielleicht aus Ihrer Umgebung wissen, leider nicht überall gleich schnell zu Veränderungen.
Nach dieser Umfrage sind wir aber in der Hoffnung bestärkt, daß Redaktion und Leserschaft der MP auch weiterhin eine verständnisvolle Gemeinschaft bilden. Bitte unterstützen Sie uns und Ihre Mitleser auch in Zukunft mit konstruktiven Hinweisen – und vor allem mit aus Ihren praktischen Erfahrungen geborenen guten Beiträgen. *Ihre Redaktion MP*

Merkmale der RISC-Architektur

Rene Steiner, Berlin

In den 60er Jahren wurde ein Entwurfskonzept für Rechnerarchitekturen gefunden, das es ermöglichte, denselben oder einen aufwärtskompatiblen Maschinenbefehlssatz in Modellen unterschiedlichster Leistungsfähigkeit anzubieten. Die Lösung bestand in einer strikten Trennung zwischen der Maschinenbefehlsschnittstelle, der logischen Organisation des Rechners und den physischen Bausteinen des Rechners: Man erfand die Mikroprogrammierung, bei der die einzelnen Maschinenbefehle prozessorabhängig in Mikroprogramme umgesetzt werden.
Voraussetzung für den Einsatz der Mikroprogrammtechnik waren die Fortschritte in der Halbleitertechnologie Anfang der 60er Jahre. Schnelle integrierte Speicherchips standen zur Verfügung, deren Kapazität aber zu klein und deren Preis zu hoch waren, um damit auch Hauptspeicher aufbauen zu können. Die Mikroprogrammspeicher waren etwa zehnmal schneller als die damals üblichen Magnetkernspeicher mit ihrer minimalen Zugriffszeit von etwa einer Mikrosekunde. Damit bestand ein ausgewogenes Verhältnis zwischen Hauptspeicher- und Mikroprogrammspeicher-Zugriffszeit, da auf einen Maschinenbefehl etwa fünf bis zehn Mikrobefehle entfielen.
Im Laufe der Entwicklung vergrößerte sich der Geschwindigkeitsunterschied zwischen den Hauptspeichern und den Mikroprogrammspeichern weiter, so daß die Reduzierung der Hauptspeicherzugriffe zu *dem* Entwurfskriterium wurde. Es lohnte sich also noch mehr, eine Sequenz von Mikroinstruktionen zu genau einem neuen Maschinenbefehl zusammenzufassen. Der Aufwand zur Implementierung des Leitwerks explodierte. In der aktuellen Diskussion bekamen diese Rechnertypen den Namen *Complex Instruction Set Computer* (CISC).
Anfang der 70er Jahre waren dann Halbleiterspeicher genügender Kapazität und Geschwindigkeit verfügbar, so daß es möglich wurde, die Hauptspeicher mit Halbleitern aufzubauen. Damit war der *prinzipielle* Geschwindigkeitsvorteil der Mikroprogrammspeicher verlorengegangen. Zusätzlich wurde in dieser Zeit auch die Cache-Speicherorganisationsform entwickelt, die die Lokalität von Daten und Programmen nutzte. Durch diese Caches wurden die Hauptspeicherzugriffe scheinbar noch schneller /1/.
Ein moderner CISC-Prozessor ist folgendermaßen charakterisiert /2/:
- Taktfrequenz > 20 MHz
- Anzahl der Pipelinestufen > 3
- Cache-Unterstützung
- mindestens eine 8-Byte-Prefetch-Queue
- Unterstützung busorientierter Multiprozessorsysteme durch die Bereitstellung von Semaphorfunktionen und der Möglichkeit der Verblockung von Buszyklen
- Implementierung von Mechanismen zur Unterstützung der Fehlertoleranz (z. B. Built-in-Self-Tests).

Die zunehmende Komplexität der Hardware und der Mikroprogramme der Rechner erhöhte nicht nur den Entwurfsaufwand überproportional, sondern jedes zusätzliche Schaltelement und jede zusätzliche Verzweigung erhöhten durch die hinzukommenden Kapazitäten die Gatterlaufzeiten. Diese Probleme forcierten die Suche nach einfacheren Hardwarestrukturen.
Die Entwicklung der Software führte dazu, daß die Rechner heute im wesentlichen mit Hochsprachen programmiert werden, das heißt, die Maschinenbefehle werden nicht mehr durch einen geschickten Programmierer sorgfältig ausgewählt, sondern ein Compiler generiert die Maschinenbefehlsfolgen. Compiler selbst sind aber oft nicht in der Lage, komplexe Befehle, die die Hardware bereitstellt, zu nutzen, da wegen der möglichst schnellen Überführung der Compiler auf unterschiedliche Rechner kaum auf die spezifischen Eigenschaften der Prozessoren eingegangen wird. Umfangreiche Analysen des durch Compiler generierten Codes ergaben dann auch die sogenannte 90:10-Regel. Diese besagt, daß 90 % der ausgeführten Befehle nur 10 % der vorhandenen CISC-Befehle nutzten; oder anders ausgedrückt: 90 % der Entwicklungskosten (im weitesten Sinne) für die gewaltigen Mikroprogramme sind nahezu umsonst.
Diese Erkenntnisse führten insbesondere bei IBM, an der Berkeley- und an der Stanford-University Anfang der 80er Jahre zur Entwicklung der RISCs (Reduced Instruction Set Computers). Ihre wesentlichsten Merkmale sind:

– *überwiegend Einzyklus-Maschinenbefehle*
Maschinenbefehle lassen sich logisch in verschiedene Phasen unterteilen; der RISC-Prozessor R2000 beispielsweise wurde in folgende unterteilt /2/ (vgl. auch Bild 1):
1. Befehl holen (IF)
2. Befehl dekodieren und Operanden von den Registern holen (RD)
3. Befehl ausführen (ALU)
4. Speicherzugriff (MEM)
5. Resultat in die Register schreiben (WB)

Werden diese logischen Phasen getrennt in der Hardware umgesetzt, ist es möglich, die

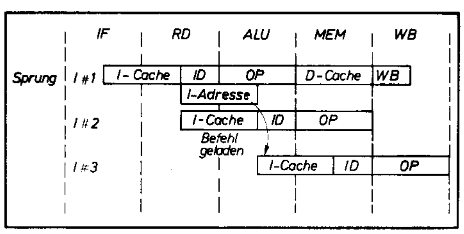


Bild 1 *Sequentielle Abarbeitung der Maschinenbefehle*

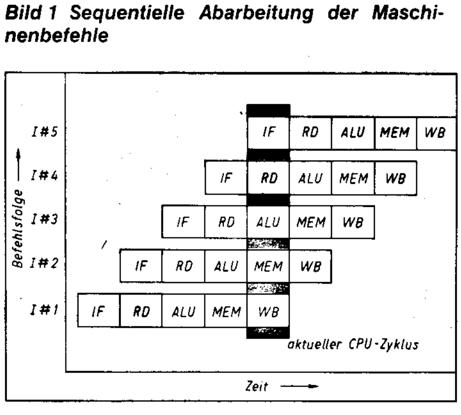


Bild 2 *Pipelineabarbeitung der Maschinenbefehle*

Befehlsausführung zu überlappen. Das bedeutet, daß der nächste Maschinenbefehl nicht erst nach der vollständigen Abarbeitung des vorhergehenden abgearbeitet wird, sondern daß sich in der Pipeline mehrere Maschinenbefehle in unterschiedlichen Bearbeitungsphasen befinden (vgl. Bild 2 für den R2000). Zwar braucht bei einer n-stufigen Pipeline die Abarbeitung eines vollständigen Maschinenbefehls n Takte, aber bei einer ständig gefüllten Pipeline wird mit jedem Takt ein Befehl abgeschlossen, so daß die resultierende Beschleunigung durch die Pipeline fast n-fach ist.

– *wenige Maschinenbefehle*
Es werden nur die Befehle bereitgestellt, die von den Compilern am häufigsten genutzt werden und die nicht zu einer exklusiven Nutzung von Hardware-Ressourcen durch einzelne Befehle führen – ausgenommen natürlich den Fall, daß dies funktional unabhängig ist. Der R2000 hat beispielsweise nur 74 Befehle. Da in einem Befehl immer beide Operanden- und das Zielregister angegeben werden (3-Adreß-Maschine), haben die RISCs eine nicht zerstörende Arbeitsweise, das heißt, keiner der Operanden wird durch das Resultat zwangsläufig überschrieben. Das wirkt sich positiv auf die Reduzierung der Hauptspeicherzugriffe aus.

– *horizontales Maschinenbefehlsformat*
Um die Dekodierung der Befehle zu vereinfachen, das heißt sie zu beschleunigen, haben alle Befehle die gleiche Wortbreite, wenige Formate, und die Felder innerhalb eines Wortes haben minimale Abängigkeiten untereinander (vgl. Tafel 1). Im Prinzip hat jede Bitstelle eine feste Bedeutung. Das erhöht zwar die Redundanz der Kodierung, hat aber den Vorteil einer einfachen Dekodierung.

Tafel 1 Befehlsformate des R2000

R-Type (Register)

31	25	20	15	10	5	0
op	rs	rt	rd	sh	fct	

I-Type (Immediate)

op	rs	rt	immediate

J-Type (Jump)

op	target

op	6-Bit-Operationscode
rs	5-Bit-Quellregisteradresse
rt	5-Bit-Quellregisteradresse oder Sprungbedingung
rd	5-Bit-Zielregisteradresse
sh	5-Bit-Shift-Amount
fct	6-Bit-Funktionsfeld
immediate	16-Bit-Sprung- oder Adreß-Offset
target	26-Bit-Sprungzieladresse

– *Load-/Store-Architektur*
Alle Datenmanipulationsbefehle können mit nur einem großen Registersatz, der mittels der Load-/Store-Befehle seinen Inhalt mit dem Hauptspeicher austauschen kann, arbeiten. Dies führt insbesondere zu einer Vereinfachung der Befehlsformate und zu einer Vereinheitlichung der Speicherzugriffe, was ein Pipelining erleichtert. Der R2000 beispielsweise hat nur eine Adressierungsart: Basisregister + 16-Bit-Offset /3/.

– *festverdrahtetes Leitwerk*
Die Steuersignalgenerierung erfolgt nicht durch ein Mikroprogramm, sondern maßgeschneidert und festverdrahtet, in der Regel durch relativ einfache und kleine PLD-Strukturen (ca. 5–10 % der Chipfläche).

– Verlagerung möglichst vieler Aufgaben in die Compiler

Die notwendigen Optimierungen erfordern, daß keine oder nur wenige Maschinenzustände für den Compiler verdeckt sind, das heißt, die Portabilität von Programmen wird vom Maschinenniveau in das Hochsprachenniveau verlagert, da Prozessor und Compiler eine Einheit bilden. Dabei verlagern sich etwas die Schwerpunkte bei der Entwicklung von Compilern. Einerseits werden sie einfacher, da:

① sich durch die Reduzierung der Befehlsanzahl und der dazugehörigen Adreßmodi die Anzahl der zu untersuchenden Alternativen verringert (z. B. muß das Quellprogramm nicht mehr auf Strukturen hin untersucht werden, die durch einen komplexen Maschinenbefehl ersetzt werden können)

② durch die einheitliche Befehlsbreite von 32 Bit Optimierungen wesentlich leichter auszuführen sind (ein gegenseitiges Austauschen und Verschieben der Befehle hat keine Speicherprobleme zur Folge)

③ im Prinzip alle Register allgemeine Register sind, die sowohl zur Adreß- und Operandenberechnung als auch zur Speicherung von Bedingungsbits genutzt werden können und somit sich eine vereinfachte Registerzuweisungsstrategie ergibt.

Andererseits verkompliziert sich der Compiler dadurch, daß die RISCs eine sehr große Anzahl von Registern haben, sowie durch das notwendige Auflösen von Pipelinekonflikten.

– reguläres Layout

Der vereinfachte und gestraffte Funktionsumfang der Chips erlaubt das Integrieren großer regulärer Strukturen, was die Nutzung neuester Halbleitertechnologien ermöglicht, da das Design und die Fehlersuche vereinfacht werden.

Die relativ tiefen Pipelines (> 4 Stufen) müssen ständig gefüllt sein, um das angestrebte Ziel von 1 IPC (Instruction per Cycle) zu erreichen. Dem stehen im wesentlichen drei Faktoren entgegen:

– Sprünge, das heißt die Lokalität des Programmablaufs, sind nicht mehr gewährleistet.

– Operationen benötigen Operanden, die noch nicht verfügbar sind, da sie noch in der Pipeline berechnet oder aus dem Speicher geholt werden.

– Unterschiedliche Befehle in der Pipeline fordern die gleichen Ressourcen an.

Gelöst werden die Probleme durch eine Kombination von Hard- und Softwaremaßnahmen. Durch einen Sprungbefehl wird normalerweise die Pipelineabarbeitung verzögert, da bei Erfüllung der Sprungbedingung erst die effektive Sprungadresse berechnet wird und dann der dort stehende Befehl geholt werden muß. In den meisten RISC-Prozessoren wird deswegen der Befehlsfluß solcherart umsortiert, daß die Sprungausführung um einen Takt verzögert wird und in dem dazwischenliegenden Takt ein NOP oder ein durch den Compiler umsortierter, semantisch sinnvoller Befehl abgearbeitet wird (für den R2000 ist dies im Bild 3 dargestellt). Auf diese Weise ist die Wahrscheinlichkeit wesentlich größer, daß die Pipeline ständig gefüllt ist und nach einem Sprung nicht neu aufgebaut werden muß.

Daß die Register bei der Ausführung einer Operation immer die aktualisierten Daten enthalten, wird entweder dem Compiler oder einer speziellen Hardware überlassen, beispielsweise dem sogenannten Scoreboard. Dieses Scoreboard ist aus dem Supercomputerbereich bekannt und wurde erstmals in der innovativen CDC 6600 angewendet. Jedem Register wird ein Bit zugeordnet, welches im gesetzten Zustand anzeigt, daß das entsprechende Register das Zielregister einer in der Pipeline befindlichen Operation ist und demzufolge zugriffsgeschützt ist, da das Register noch nicht aktualisiert wurde. In bestimmten Fällen wird das Register-bypassing verwendet. Dabei wird ein Resultat, welches in der nachfolgenden Operation als Operand benötigt wird, sowohl entsprechend der Befehlssemantik in das Zielregister geschrieben als auch über einen Extrapfad der ALU sofort als Operand zur Verfügung gestellt.

Da die Speicherzugriffszeiten in der Regel länger als die Taktzyklen der Prozessoren sind, ist die Nutzung der zu ladenden Daten unmittelbar nach dem Ladebefehl nicht möglich. Also erfordert die folgende Befehlssequenz

SUB R23, R8, R9
LOAD adr, R2
ADD R1, R2, R3

die Einführung einer entsprechenden Anzahl von NOPs zwischen dem Ladebefehl und dem Additionsbefehl, damit das Datum im Register R2 beim Lesen durch den Additionsbefehl vorhanden ist. Bei der Verwendung eines Caches zwischen dem Hauptspeicher und dem Prozessor könnte sich dann ergeben:

SUB R23, R8, R9
LOAD adr, R2
NOP
ADD R1, R2, R3

Ein optimierender Compiler sortiert den Code so um, daß der Ladebefehl soweit wie möglich vorgezogen wird, damit zwischen der Ausführung des Ladebefehls und des Additionsbefehls sinnvolle Befehle ausgeführt werden. In diesem konkreten Fall ist die Verschiebungsweite davon abhängig, ob die effektive Adresse des Ladebefehls berechnet ist und in einem Register steht und ob die Daten des Registers R2 schon aktualisiert werden dürfen, da der alte Registerinhalt nicht mehr benötigt wird. Durch die nicht zerstörende Arbeitsweise (3-Adreß-Maschine) sind solche Befehlsverschiebungen leichter möglich, da die semantischen Abhängigkeiten der Befehle untereinander reduziert werden. Damit ergibt sich:

LOAD adr, R2
SUB R23, R8, R9
ADD R1, R2, R3

(Vgl. dazu auch Bild 4).

Ein weiterer Problemkreis bei der Gewährleistung von 1 IPC sind die Geschwindigkeitsanforderungen an den Hauptspeicher. Zwar sind prinzipiell Speicherchips vorhanden, die die erforderlichen Geschwindigkeiten sicherstellen können, aber Kostengründe sprechen

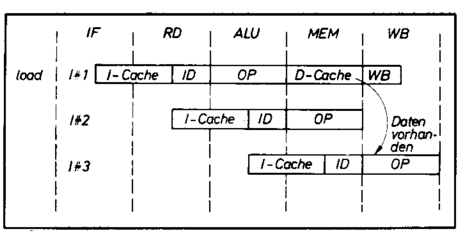


Bild 3 Verzögerter Sprung im R2000

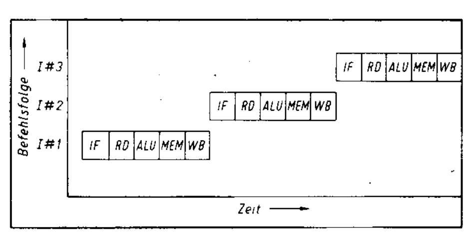


Bild 4 Verzögertes Laden im R2000

Glossar

Cache
Der Cache ist ein im Verhältnis zum Hauptspeicher sehr schneller Speicher. In ihm ist ein auswechselbarer Ausschnitt des Hauptspeicherinhaltes kopiert, mit dem die CPU arbeitet (siehe auch /6/).

Compiler
Der Compiler ist ein Programm, das ein in einer Hochsprache (Basic, C, Fortran ...) geschriebenes Quellprogramm in die entsprechenden Maschinenbefehle übersetzt. Dabei wird das Quellprogramm auf Syntaxfehler überprüft und das Maschinenprogramm optimiert.

Kontextwechsel
Bei einem Kontextwechsel werden alle Register, die den prozessorinternen Zustand für das aktuelle Programm kennzeichnen, in den Hauptspeicher gerettet, damit ein anderes Programm mit seinem Kontext in den Prozessor geladen werden kann. Somit ist es später möglich, das „gerettete" Programm an der gleichen Stelle wieder fortzusetzen.

Maschinenbefehl
Der Maschinenbefehl ist ein in Binärform dargestellter Befehl, der vom jeweiligen Zielprozessor unmittelbar ausgeführt werden kann.

Mikroprogrammtechnik
Mittels der Mikroprogrammtechnik werden innerhalb des Prozessors die nach außen unteilbaren Maschinenbefehle in Sequenzen von Mikrobefehlen zerlegt. Diese Mikrobefehle werden aus dem internen Mikrobefehlsspeicher geholt und steuern die einzelnen Funktionseinheiten des Prozessors an. In der Regel sind die Sequenzen für unterschiedliche Maschinenbefehle unterschiedlich lang, bzw. die Länge der Sequenz hängt oft von Parametern des Maschinenbefehls (z. B. Laufvariablen) ab.

Portabilität
Die Portabilität ist ein Softwaremerkmal, welches ausdrückt, daß ein Programm nicht nur auf einer Maschine ausführbar ist. Hochsprachen sind portabler als Maschinenprogramme. Mit wachsender Portabilität von Programmen sinkt oft deren Effizienz.

Prefetch-Queue
Die im sequentiellen Ablauf dem aktuellen Befehl folgenden Befehle werden in die CPU geladen, ohne daß sie sofort abgearbeitet werden. Gespeichert werden sie in der Prefetch-Queue. Dies macht den Hauptspeicherzugriff aus der CPU-Sicht scheinbar regulärer, da solche Probleme wie das Auffrischen von DRAMs oder die unterschiedliche Länge von Befehlen bei CISC-Architekturen in ihren Auswirkungen nicht mehr so gravierend sind.

Tag-Befehle
Ein Tag ist eine zusätzliche Information (z. B. Typkennung) zu Daten oder Instruktionen. Oft werden sie mittels weniger Bits innerhalb der Daten hardwaremäßig realisiert. Tag-Befehle werten diese Informationen aus.

Erfolge für RISC-Architektur

Anfänglich in Spezialrechnern angewendet, ist das RISC-Konzept seit 1989 bei allen großen Rechnerproduzenten in die Architektur einbezogen worden. Heute besitzt etwa jede zehnte ausgelieferte 32-Bit-CPU RISC-Merkmale, aber 1992 soll nach Dataquest-Berechnungen dieser Anteil 30 % betragen.
RISC-Aktivitäten unternehmen die Firmen Acorn, Ridge, Fairchild, Pyramid, MIPS Computer-Systems, Hewlett-Packard, Inmos, DEC und IBM. Die größten Marktchancen werden gegenwärtig aber der Firma MIPS Computer-Systems zugebilligt. Diese Firma hat sich durch großzügige Lizenzvergabe einen großen Marktanteil gesichert, ohne selbst zu fertigen. Von etwa 100 Mio Dollar Umsatz kamen 1989 35 % aus solchen Kooperationen. Siemens, NEC und LSI fertigen die MIPS-Prozessoren selbst; Prime, Silicon Graphics, Tandem, DEC, IN2, PCS, Sony, Bull und Nixdorf verwenden den MIPS-Chip.
Eine ähnliche Strategie verfolgt die Firma Sun. Zu den Lizenznehmern dieser Firma zählen: Mizar, Philips, Definicon, Datatech, Tatung, Toshiba, ICL, Rank Xerox, Solbourne, Arix und Prisma.
Die Firma Motorola hat den Prozessor 88000 entwickelt, den Data General, Tektronix, Sanyo/Icon, NCR und Unisys produzieren wollen. Insgesamt werden jedoch von Motorola keine größeren Aktivitäten auf dem RISC-Sektor beobachtet.
Intel hat mit dem i860 erst relativ spät einen RISC-Prozessor vorgestellt. Die Fachleute vermuten, daß durch das späte Auftreten kaum noch ein Kundenkreis gewonnen werden kann.
IBM wartete mit einer eigenen RISC-Lösung für die RS/6000-Familie auf. Die erhöhte Rechenleistung konnte bei Workstations direkt umgesetzt werden.
Der Sun-Chip erreichte bereits 65 Mio Instruktionen pro Sekunde. Die Firma MIPS will 55 Mio Instruktionen pro Sekunde erreichen. Data General arbeitet an einer 100-Mips-Version und Nokia Data an einer 1000-Mips-Version.
Da fast alle RISC-Systeme unter Steuerung des Betriebssystems Unix laufen, kann die Software günstig portiert werden. *Wie*

gegen ihren massenhaften Einsatz. Gelöst wird dieses Problem durch die verstärkte Einführung der Harvard-Architektur, die Verwendung von verschränkten Speichern und Caches sowie durch große Registersätze. Bei den Registersätzen gibt es im wesentlichen drei Ansätze:

① beispielsweise 32 allgemeine Register, wobei eines meistens auf 0 geschaltet ist. Die Verwaltung wird entweder durch Scoreboards oder durch die Software übernommen (vgl. Tafel 2).

② Ausgehend davon, daß in einem typischen C- oder Pascal-Programm zirka 40 % der Speicherzugriffe durch Calls oder Returns verursacht wird, werden mehrere Registerfenster, die in einem überlappenden Registersatz angeordnet sind, integriert. Jedes dieser Fenster besteht aus drei Teilen: Input-Register, Output-Register und lokale Register. Dabei sind die Output-Register des aktuellen Fensters mit den Input-Registern des folgenden Fensters physisch identisch. Eine Prozedur kann nur auf ein Fenster und auf die zusätzlich vorhandenen globalen Register zugreifen. Die Überlappung der Registerfenster ermöglicht eine Parameterübergabe zwischen Programmteilen ohne zeitaufwendige Stackoperationen (vgl. Bild 5). So werden bei Unterprogramm-Aufrufen und ähnlichem nur die Registerfenster umgeschaltet. Durch die wesentliche Reduzierung der Speicherzugriffe haben diese großen Registersätze einen cacheartigen Charakter. Vorteilhaft ist eine solche Organisation auch für den Compiler, da die Registerzuweisungsstrategien wesentlich einfacher werden. In der Regel sind zusätzlich mehr als 8 globale Register vorgesehen, auf die alle Prozeduren zugreifen können. Nachteilig bei dieser Lösung ist, daß
– beim Kontextwechsel alle aktiven Registerwindows gerettet werden müssen
– durch die feste Fenstergröße die Registeranzahl oft zu groß oder zu klein ist
– die kapazitive Belastung der Registerbusse sowie der Platzbedarf zu hoch sind.
Um diese Nachteile zu vermindern, sind von verschiedenen Entwicklern andere Architekturen, zum Beispiel mit variabler Fenstergröße, die sich aber aufgrund des damit verbundenen höheren Steueraufwandes bisher nicht durchsetzen konnten, vorgeschlagen worden.

③ Es sind sehr viele Register vorhanden (ca. 200), und diese sind zusätzlich zu ihrer Organisation als Register noch wie ein Stack organisiert und ermöglichen in demselben physischen Speicherraum innerhalb des Prozessors sowohl die Registerfunktion als auch den obersten Ausschnitt des Laufzeit-Stacks. Das verkompliziert zwar das Leitwerk, hat aber den Vorteil, daß
– nur die benötigten Register belegt werden
– die Übergabe von Parametern genauso leicht möglich ist, wie bei den überlappenden Registersätzen
– ein Kontextwechsel sehr schnell sein kann.
Nachteilig ist natürlich auch hier die hohe kapazitive Belastung.

Tafel 2 Registersatz des R2000

General Purpose Register
R 0
R 1
R 2
R 29
R 30
R 31

Multiple/Divide Register
HI
LO

Program Counter
PC

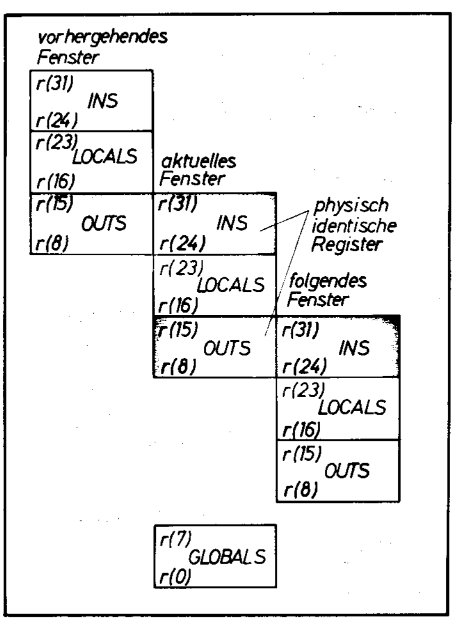


Bild 5 Registerfenster des SPARC-Prozessors

Die überraschenden Ergebnisse und umfangreichen Arbeiten der Berkeley-Gruppe und auch der Stanford-Gruppe zur RISC-Architektur starteten unter den Produzenten einen Wettlauf zur Entwicklung von RISC-Workstation-Mikroprozessoren. Mit der Entwicklung von RISC-Prozessoren begannen nahezu alle namhaften Produzenten. Dazu gehören Sun Microsystems mit dem SPARC (Scalable Processor ARChitecture), Motorola mit dem M88000, Hewlett-Packard mit der High-Precision Architecture, MIPS Computer Systems mit dem R2000 und dem kompatiblen R3000, AMD mit dem Am29000, Intergraph Advanced Processor Division mit dem Clipper und Intel mit dem 80960 und dem 80860 (siehe auch /4/).
Um den Anforderungen an Arbeitsstationen gerecht zu werden, mußten noch einige Lücken gegenüber den Entwicklungen an den Universitäten (wie Gleitkomma-Unterstützung, Speicherverwaltung, Cache-Design u. a.) von den Produzenten geschlossen werden. So hat die Beschränkung der Maschinenbefehle zur Folge, daß zum Teil häufig geforderte komplexere Befehle (z. B. die Befehle der Gleitkomma-Arithmetik) durch Softwareroutinen emuliert werden müssen. Um solche Befehle auch in der RISC-Technik schnell in der Hardware zu ermöglichen, wird das Koprozessorkonzept angewendet. Dies eröffnet die Möglichkeit, in die Befehlssätze Befehle aufzunehmen, die für Spezialanwendungen gedacht sind. So hat beispielsweise der i80860 zusätzliche Grafikbefehle, der M88000 dagegen mehrere Bitfeld-Befehle, und im SPARC sind Tag-Befehle integriert.
Im kommerziellen Bereich haben sich die RISCs vor allem auf dem Workstation-Markt durchgesetzt. Es gibt heute de facto keinen führenden Computerhersteller, der keine Workstations mit RISCs anbieten würde. Dabei werden sogar solche marktentscheidenden Merkmale wie die Softwarekompatibilität zu den Vorgängern aufgegeben. Spektakulär in dieser Beziehung war die Vorstellung der DECstation 3100 der Firma Digital Equipment, da auf ihr nur noch das Unix-kompatible Betriebssystem ULTRIX läuft.
Auf dem PC-Markt konnten die RISCs bisher noch keine nennenswerten Anteile erobern. Wahrscheinlich ist die für RISC-Prozessoren notwendige Umgebung, insbesondere die schnellen und großen Speicher, noch zu teuer für PCs. Aber auch hier kommt der Markt in Bewegung, wie die Ankündigung eines auf einem SPARC aufbauenden Laptops der Firma Toshiba zeigt /5/. Dieser Laptop hat 8 MByte Hauptspeicher, eine 180-MByte-Festplatte und eine Bildschirmauflösung von 1152 × 900 Bildpunkten.

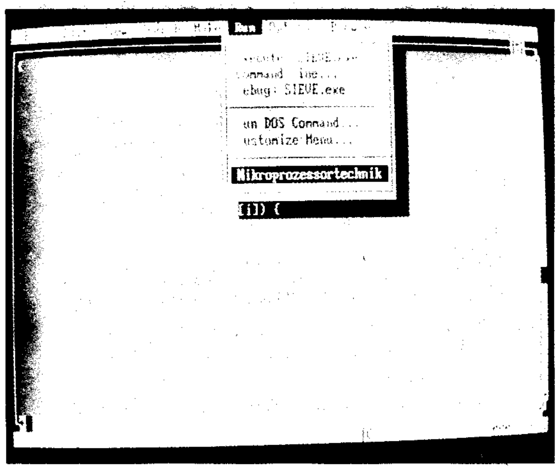
Fortsetzung auf Seite 9

kennengelernt

Microsoft C 6.0 PDS

Uwe Schulze, Berlin

Lange Zeit fuhr Microsoft zweigleisig. Auf der einen Seite die großen und zum Teil recht betagten Optimierungscompiler, auf der anderen die Quick-Serie, für Einsteiger gedacht und inzwischen den Kinderschuhen entwachsen. Die neuen Professional Development Systems (PDS) sollen die Vorteile beider Linien in sich vereinen. Außerdem werden die verschiedenen Programmiersprachen in einer Entwicklungsumgebung – der Programmers Workbench (PWB) – zusammengeführt. C 6.0 und Basic 7.0 sind bereits verfügbar, Pascal soll folgen. MP stellt Ihnen den neuen C-Compiler näher vor, der in einer Beta-Version vorab zur Verfügung stand.



Installation und Editor

Die Idee ist einfach: Die Leistungsfähigkeit des Optimierungscompilers 5.1 wird mit dem Komfort von Quick-C kombiniert. Außerdem führt die Entwicklungsumgebung alle Tools (u. a. die Make-Funktion und den Stand-Alone-Debugger Code View) zusammen und sieht die Einbindung der anderen Sprachen (Assembler, Basic, Pascal) vor. Zunächst aber fällt der nicht unerhebliche Umfang gelieferter Software auf: 7 Disketten mit über 8 MByte Programmen, zum Teil in komprimierter Form. Die Installation, die zum Entpacken unumgänglich ist, verläuft im Gegensatz zu den sonstigen Gepflogenheiten bei Microsoft – wo ohne Maus und Fenster nichts geht – im zeilenweisen Dialog ab und dauert etwa eine Stunde, davon die Hälfte zum Entpacken und Kopieren, die andere Hälfte zum Erstellen der Bibliotheken. Eine vollständige DOS-Version für alle Speichermodelle belegt anschließend 8 MByte auf der Festplatte. Sie enthält die neue Entwicklungsumgebung (Programmers Workbench, kurz PWB genannt), Compiler, zwei Linker, Bibliotheken, Quellen von Bibliotheksfunktionen, Beispiele, umfangreiche Hilfstexte und den Debugger CodeView in der Version 3.0. Alternativ können auch die OS/2-Bibliotheken installiert werden.

Minimale Systemanforderungen sind 512 KByte Hauptspeicher und 5 MByte freie Plattenkapazität, für die praktische Arbeit sind aber 10 MByte externer und 640 KByte Hauptspeicher plus EMS oder den häufig vorhandenen 384 KByte Extended Memory zu empfehlen.

Der Editor entspricht im wesentlichen dem von Quick-C. Es handelt sich um einen Multi-window-Editor, mit dem mehrere Texte parallel bearbeitet werden können, während – auf Wunsch – auch noch Hilfe- und Ausgabefenster sichtbar sind. Alle Fenster sind in Größe und Lage veränderbar; für eine effektive Arbeit wird eine Maus benötigt. Der Editor ist frei konfigurierbar, was insbesondere für Programmierer interessant sein wird, die nicht von Wordstar kommen. Farben und Muster sind (wie in Quick-C) interaktiv einstellbar. Weitere leistungsfähige Features stellen die einfach zu bedienende *Undo*-Funktion dar, mit der die letzten Aktionen annulliert und auch wieder gültig gemacht werden können, sowie ein Makro-Rekorder und eine Makro-Sprache, zur wiederholten Abarbeitung von Befehlsfolgen. Zum Lernen kann man einen Blick in die Datei Tools.ini werfen, wo die Standardeinstellungen als Makro eingetragen sind. In puncto Makrosprache bieten spezielle Programmeditoren (z. B. Brief) zwar mehr, doch können Routinen zur Erweiterung des Editors auch in C geschrieben werden, was keine Wünsche offen läßt. Beim Verlassen der Workbench werden alle Einstellungen automatisch gespeichert. Das gilt auch für die anderen Tools (Debugger, Make-Utility), was zeigt, wie die Workbench das Zusammenspiel der einzelnen Programme unterstützt.

Die Optimierung

Beim Kompilieren und Linken wird die jeweilige Kommandozeile mit allen Parametern angezeigt, so daß die eingestellten Schalter überprüfbar sind. Wahlweise kann eine inkrementelle Übersetzung veranlaßt werden (die nur veränderte Funktionen berücksichtigt) oder aber – zum Abschluß der Programmentwicklung – die Optimierung des Gesamtwerkes. Diese Trennung nahmen viele Programmierer auch bisher schon vor, indem sie mit Quick-C entwickelten und das Ergebnis mit MSC 5.1 übersetzten. Aber C 6.0, das ist nicht nur eine neue Umgebung. Vielmehr wurde die bisher schon gute Optimierung weiter verfeinert, so daß Microsoft C die Spitzenstellung in seiner Klasse, was die Kodequalität angeht, gegen die Konkurrenten Turbo C und Lattice C weiter ausbauen dürfte. Der Programmierer erhält mehr Kontrolle über die Optimierung, da zum einen mehr Schalter zur Verfügung stehen, diese zum anderen gezielt (nämlich lokal in Blöcken und Funktionen) eingesetzt werden können. Bedeutung erlangt diese Möglichkeit im Zusammenhang mit dem Debugging und der Vermeidung von Nebeneffekten bei bestimmten Programmiertricks. Am einfachsten ist die Einstellung der Optionen über das *Option*-Menü der Workbench möglich, allerdings gelten sie dann für den gesamten Modul. Den lokalen Einsatz verfügt man mit dem neuen Progma optimize zur Optimierung des Registereinsatzes:

#pragma optimize ("e",on)

Der Compiler hält in diesem Falle die am häufigsten benutzen Variablen in Registern, was den Einsatz der Speicherklasse *register* überflüssig macht, so daß sie beim Setzen des Schalters auch ignoriert wird. Zentraler Punkt ist die Schleifenoptimierung, weil hier besonders viel Laufzeit eingespart werden kann. Sie wird in drei Stufen offeriert, aber nicht etwa aus Gründen der Übersetzungszeit. Vielmehr wird eine *extreme* und eine *sichere* sowie überhaupt *keine* Optimierung unterschieden, wobei erstere bei Sonderfällen zu Problemen führen kann. Für jede der drei Einstellungen existiert ein eigener Schalter. Einige Verbesserungen nimmt der Compiler stets vor, wenn die Optimierung nicht grundsätzlich abgeschaltet wurde (Disable Optimization im Option-Menü). Dazu gehören die Berechnung von konstanten Ausdrükken beim Kompilieren und die automatische Einführung temporärer Variablen für mehrfach benutzte Ausdrücke. Weiterhin kann auch auf die Laufzeitoptimierung zugunsten einer Kodeoptimierung verzichtet werden. Eine Reihe von Programmen läuft etwas schneller als unter MSC 5.1. Bei vollständiger Abwärtskompatibilität kann man sich das zunutze machen, indem man seine alten Programme neu übersetzt. Die Übersetzung wird von der integrierten Make-Funktion gesteuert, die alle Abhängigkeiten überprüft. Zum Erstellen von Projekten existiert eine Dialogbox, woraus dann automatisch das kryptische Make-File erzeugt wird (wie auch schon bei Quick-C). Weiterhin ist aber die Arbeit auf Kommandoebene durch direkten Aufruf von Compiler (Cl.exe) und Linker Ilink (inkrementeller Linker) oder Link (segment executable Linker) möglich.

Microsoft C 6.0 unterstützt endlich auch das Tiny-Speichermodell, so daß COM-Files (etwa für residente Programme) erstellt werden können. Der Long-double-Support entspricht jetzt der Breite des Koprozessors (80 Bit, bisher 64 Bit) und das auch als Emulation. Dazu wurden die arithmetischen Funktionen komplett überarbeitet.

Neu eingeführt sind sogenannte Based-Pointer, die mit 16 Bit (wie Near-Pointer), aber bezogen auf ein Basissegment auch Adressen über die Grenzen des 64 KByte-Segments hinaus ansprechen können.

Inline-Statements werden vom Compiler ohne separaten Assemblerpaß behandelt. Als weitere Neuerung ist die Parameterübergabe an Funktionen in Registern möglich, wenn diese als **_fastcall** deklariert sind – aufgrund der beschränkten Anzahl von Registern aber nur bis zu drei Parameter.

Der Entwicklungskomfort

Während der Programmentwicklung braucht die Workbench nicht verlassen zu werden, da Linker, Debugger oder andere Programme (etwa ein Profiler) von hier aus aufgerufen werden können. Für häufig ausgeführte Aktionen lassen sich in einer Dialogbox auch die Pull-Down-Menüs um zusätzliche Kommandos erweitern (s. Bild), einschließlich Pfad- und Parameterangaben. Weiterhin sind Aktionen in Abhängigkeit von der Dateierweiterung einstellbar, was zum Beispiel zum Aufruf fremder Compiler oder des Assemblers nutzbar ist.

Ein Novum, das seinesgleichen sucht, ist der Source-Browser, mit dem der Programmierer seinen Quelltext überwachen kann. Dazu gehört das Auffinden der Definitionen von Variablen und Funktionen bzw. aller Aufrufe einer Funktion. Ferner können alle Referenzen auf eine Variable gefunden werden. Auch die Bestimmung der Abhängigkeit bei mehreren Modulen ist möglich.

Gegenüber Quick-C wurde das Hilfssystem entscheidend erweitert. Es ist in einem halben Dutzend separater Dateien untergebracht und läßt sich zu jeder Zeit durch den rechten Mausklick aktivieren. Die angestrebte offene Architektur zeigt sich hier daran, daß der Programmierer die Hilfen selbst erweitern kann, sei es als eigene Gedächtnisstütze oder zur Arbeit im Team. Ebenfalls überarbeitet wurde der CodeView-Debugger. Sofern Extended oder Expanded Memory vorhanden ist, kann er sich fast vollständig auslagern und damit alle Programme debuggen, die in den Hauptspeicher passen.

Die Kontrolle komplexer Datenstrukturen wurde verbessert, und über History-Sessions kann – wie beim Turbo-Debugger – nach einer Quelltextänderung und Neuübersetzung die gleiche Einstellung aller Parameter (Breakpoints, Registerbelegungen etc.) erreicht werden.

Microsoft C 6.0 bietet umfassende OS/2-Unterstützung, wie beispielsweise zur Entwicklung von DLL- (Dynamically Linked Libraries) und Multithread-Programmen, und auch das neue High Performance File System (HPFS) von OS/2 kann genutzt werden. Für viele ist das zwar noch Zukunftsmusik, aber auch eine Sicherung der Investitionen.

Es soll aber auch nicht verschwiegen werden, wo die Grenzen der Workbench-Lösung liegen: Unangenehm machen sich die permanenten Plattenzugriffe zum Nachladen der einzelnen Programme (Compiler, Linker, Hilfen und Debugger) sowie zum Laden und Speichern der diversen Konfigurations-, Sitzungs- und Makefiles bemerkbar. Dem Hinweis etwa, daß zu einem Item kein Hilfstext vorhanden ist, geht eine Suche auf der Platte voraus, die mehrere Sekunden dauert. Man kann sich des Eindrucks nicht erwehren, daß es sich Microsoft mit dem Zusammenfassen von Einzelprogrammen unter einer Oberfläche etwas leicht gemacht hat und daß bei Entwicklungsumgebungen mit echt integriertem Compiler schon Besseres zu sehen war. Der Hersteller ist sich des Übergangscharakters seiner Lösung wohl bewußt, denn in Zukunft wird man voll auf die Arbeit unter Windows und dem OS/2-Presentation Manager setzen. Meine vom damals überzeugenden Turbo-Pascal her begründete Euphorie über Compiler mit integrierter Nutzeroberfläche hat sich aber wieder gelegt.

Darf's auch ein 486er sein?

Uwe Schulze, Christian Hanisch

***Bild 1** Der LGI 486. Das Gehäuse entspricht dem des 386ers (bei dem von uns getesteten Gerät sogar mit dem alten Schildchen)*

Alle bedeutenden PC-Hersteller runden ihre Produktpalette nach oben mit einem 486er ab. Aber lohnt sich der Einsatz schon, und welche Vorteile bringt er? MP hat für Sie den LGI 486 von Lucky Goldstar International unter die Lupe genommen.

Der i80486 ist ebenso wie sein Vorgänger (der i80386) ein 32-Bit-Prozessor, besitzt aber auf dem Chip bereits eine Arithmetikeinheit (die den 80387 ersetzt), eine Speicherverwaltungseinheit (MMU) und 8 KByte Cachespeicher. Damit reduziert sich zum einen der Aufwand für den Bau des Motherboards, zum anderen wird die Geschwindigkeit deutlich erhöht. Ein 80486-Prozessor soll etwa 3mal schneller sein als sein Vorgänger bei gleicher Taktfrequenz. Ursache dafür ist neben der Integration mehrerer Bausteine auf dem Chip die Veränderung der Befehlsausführung. Häufig benötigte Befehle (zum Beispiel Ladebefehle) werden ohne Microcode in möglichst wenig Taktzyklen ausgeführt. Der 80486 weist damit Eigenschaften von RISC-Prozessoren auf. Gegenüber dem Vorgänger wurde auch der Adreßraum vergrößert, physisch auf 4 Gigabyte und virtuell auf 64 Terabyte! Eine genaue Beschreibung der Hardwareeigenschaften finden Sie in MP 9/89, Seite 278.

Die Leistungsfähigkeit eines PCs hängt aber bei weitem nicht nur vom Prozessor ab. Vielmehr sollte Wert auf den eingesetzten Bus-Standard (ISA, EISA oder MCA), die Geschwindigkeit der Festplatte (mittlere Zugriffszeit, Transferrate), die Zugriffszeit der RAM-Bausteine (bedingt eventuell Waitstates) und die Grafikkarte (8 Bit oder 16 Bit) gelegt werden. Um den enormen Preissprung zwischen 386 und 486 auszugleichen, liefern einige Hersteller ihre Geräte mit einer nur kurios zu nennenden Konfiguration aus: beispielsweise nur 512 KByte RAM oder eine 45-ms-Festplatte. In diesem Falle werden Sie aber nicht in den Genuß der Vorteile des 80486 kommen. Ein gut konfigurierter 386er kann hier für den gleichen Preis mehr leisten.

Potentielle Einsatzgebiete für 486-PCs sind Server in LANs, OS/2, Unix oder multitaskingfähige DOS-Derivate (DesqView, VM/386, PC-MOS), also alles speicher- und plattenzugriffsintensive Anwendungen, kein Number-Crunching. Nichtsdestotrotz wird aber ein originales MS-DOS 3.3 ohne Probleme von der Diskette gebootet. Ob auch im CAD-Bereich Marktanteile gegenüber den 68000-basierten Workstations gutgemacht werden können, bleibt abzuwarten. Bei der Prüfung der Gesamtausstattung sind insbesondere folgende Punkte zu bedenken:

Findet noch der alte AT-Bus Verwendung (auch ISA genannt), dann können alte Karten weiterhin genutzt werden. Dieser Aspekt dürfte aber bei der Anschaffung eines 486ers nicht im Vordergrund stehen. Um die Leistungsfähigkeit des Prozessors wirklich auszureizen, stehen entweder der von IBM entwickelte Microchannel (MCA) zur Verfügung, den aufgrund der Lizenzbedingungen nur wenige Firmen anbieten (u. a. NCR) oder der weiter verbreitete EISA-Bus, in dessen Steckverbinder ebenfalls auch alte AT-Karten passen.

Besonderer Wert sollte auf die Geschwindigkeit der Festplatte gelegt werden, denn sie bestimmt das Zeitverhalten des Gesamtsystems maßgeblich. 28 ms als mittlere Zugriffszeit sind Norm; es darf aber auch etwas schneller sein (eine 18-ms-Festplatte macht sich recht positiv bemerkbar). Aussagekräftiger als die mittlere Zugriffszeit ist häufig die Datentransferrate, da sie auch den Interleave oder Cache-Speicher des Controllers berücksichtigt. Sie sollte über 400 KByte pro Sekunde liegen; schnelle Festplatten schaffen 600 KByte pro Sekunde und mehr. Die Geschwindigkeit der Festplatte gewinnt noch an Gewicht dadurch, daß neuere Software (MS-Windows, OS/2) mit dynamischer Speicherverwaltung arbeitet und ständig Speicherseiten auf die Festplatte auslagert (Swapping).

Wieviel RAM benötigt wird, hängt von den Anwendungen ab; die Chance der Aufrüstung auf 8 MByte (besser noch auf 16 MByte) sollte man sich aber offenlassen. Durch das erwähnte Swapping hat die Speichergröße auch Einfluß auf die Arbeitsgeschwindigkeit: Je mehr Hauptspeicher vorhanden ist, desto weniger oft braucht aus- und eingelagert zu werden. Ein weiterer Geschwindigkeitsschub kann durch Parallelarbeit mehrerer Prozes-

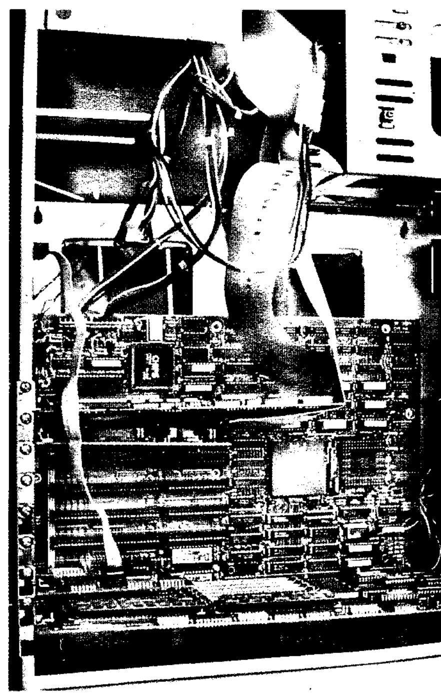


***Bild 2** Das Innenleben des LGI 486 – ausreichend Platz für die Nachrüstung weiterer Laufwerke*

Tafel 1 Leistungsvergleich von LGI 486 und LGI 286S

	LGI 486	LGI 286S
Taktfrequenz nominell	24,13 MHz (Checkit) 27,17 MHz (Chomptest) 25 MHz	16,02 MHz (Checkit) 16,18 MHz (Comptest) 16 MHz
Dhrystones	11834	3233
Whetstones	4068 KiloWs	61,6 KiloWs
Primzahlensieb	2,09 s	12,03 s
Festplatte mittlere Zugriffszeit Transferrate	SCSI 80 MByte 54 Sektoren/Spur 3 Köpfe 1012 Zylinder 24,4 ms (Checkit) 26,3 ms (Coretest) 397,3 KByte/s (Checkit) 435,2 KByte/s (Coretest)	 42 MByte 17 Sektoren/Spur 6 Köpfe 819 Zylinder 32,1 ms (Checkit) 31,1 ms (Coretest) 148,8 KByte/s (Checkit) 239,1 KByte/s (Coretest)

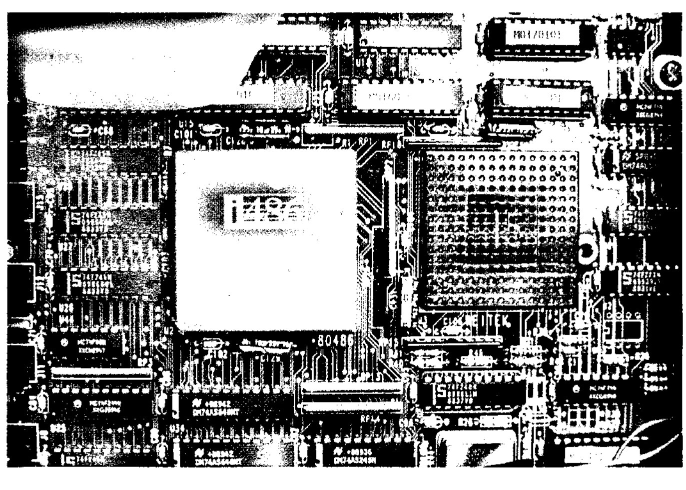


***Bild 3** Der Prozessor i486 und der Sockel für den Weitek-Koprozessor 4167*

soren erreicht werden, wofür der 80486 auch ausgelegt ist und was im Compaq Systempro schon angeboten wird.

Der LGI 486

Der LGI 486 steckt in einem großen Tower-Gehäuse, das einen weiteren Ausbau gestattet. Das Motherboard entspricht aber in den Abmaßen denen eines Desktop-Gerätes, so daß viel Platz für weitere Festplatten oder Laufwerke bleibt (Bild 1). Der Chipsatz unterscheidet sich nicht von dem des ATs: 2 Interrupt-Controller (8259A-kompatibel), 1 Timer/Counter (8254-2-kompatibel), 1 Real-Time-Uhr (MC146818A-kompatibel), 114 Byte CMOS-RAM, 1 Tastaturcontroller (i8742) und 7 DMA-Kanäle (i8237A).

Neben dem Prozessor befindet sich ein Sokkel für den Weitek-Koprozessor WTL 4167 (25 MHz.), der für rechenintensive Anwendungen eine weitere Geschwindigkeitssteigerung um den Faktor 3 bis 5 bringen soll (Bild 2). Da der 80486 aber bereits eine Arithmetikeinheit enthält, wird sich diese Investition nur für wirklich rechenintensive Aufgaben lohnen.

Das Motherboard verfügt über 6 lange und 2 kurze Steckplätze, wovon die VGA-Karte einen langen, eine I/O- und eine Controllerkarte jeweils einen kurzen Slot belegen. Verwendet wird der AT-Bus, in Zukunft wird Lucky Goldstar aber dem EISA-Konzept folgen. Der Hauptspeicher befindet sich nicht auf dem Motherboard, sondern kann als separate Speicherkarte in einen dafür vorgesehenen 32-Bit-Slot gesteckt werden. Zwei verschiedene Speicherboards werden angeboten: Ein 5-MByte-Board mit fünf Bänken zu je 1 MByte (8 × 256 K × 4 und 4 × 256 K × 1) und einer Zugriffszeit von 100 ms. Das 16-MByte-Board (das auch der MP mit 4 MByte zur Verfügung stand) wird mit SIMM-Modulen zu je 256 × 9 oder 1 × 9 bestückt. Jede Bank ist mit 4 SIMMs belegt, so daß es im ersten Fall 1 MByte, im zweiten 4 MByte enthält. Der Speicher zwischen 640 KByte und 1 MByte kann zur Beschleunigung der ROM-Zugriffe als Shadow-RAM verwendet werden; der Adreßraum oberhalb von 1 MByte steht Extended-Memory-Anwendungen zur Verfügung, kann aber – da der Prozessor das unterstützt – über einen zusätzlichen Treiber auch als Expanded Memory (EMS) genutzt werden, wie ihn viele DOS-Programme fordern.

Der LGI 486 besitzt eine kleine Bedienkonsole, auf der sich neben dem Reset und der Taktumschaltung auch ein Paßwort einstellen läßt, das im angeschlossenen CMOS-RAM aufbewahrt wird. Das vermittelt ein hohes Maß an Datensicherheit; ob aber der im Gegensatz zu einem konventionellen Schlüssel kompliziertere Paßwort-Mechanismus auch ein Plus an Bedienkomfort bietet, ist zweifelhaft.

Leistungsvergleich

Über Sinn und Unsinn von Benchmarks läßt sich ganz sicher streiten, irgendeine Möglichkeit des Vergleiches muß man aber finden, und so haben wir die gängigen Testprogramme benutzt und den LGI 486 einem AT aus gleichem Hause gegenübergestellt (Tafel 1). Da die Ergebnisse verschiedener Programme ein wenig voneinander abweichen, wurde jeweils der kleinste und der größte Wert aufgeführt und in Klammern das benutzte Programm vermerkt, so daß die Angaben nachvollziehbar bleiben. Die Werte für

Tafel 2 Praxistest: Laufzeitvergleich eines Anwendungsprogramms aus der Wasserwirtschaft auf verschiedenen Rechnern

Rechnertyp	Abarbeitungszeit
IBM AT-03 (6 MHz, 80287)	1:24:00 h
Noname AT-386 (16 MHz, 80387)	12:20 min
robotron K 1840	10:30 min
Peacock Tower-AT-386 (33 MHz, 80387)	5:40 min
LGI 486	3:34 min
	2:35 min*

* Rechenzeit bei Unterdrückung aller Statusmeldungen auf dem Bildschirm

die Festplatte hängen zwar nicht vom Prozessor ab, wurden aber der Vollständigkeit halber mit aufgeführt. Es zeigt sich, daß ein 486er eine zum Teil erhebliche Geschwindigkeitssteigerung bringt. Das Primzahlensieb arbeitet er sogar mehr als doppelt so schnell ab wie ein 386er mit gleicher Taktfrequenz, der 5,60 Sekunden braucht (Vergleichen Sie auch andere Prozessoren in MP 4/90, Seite 104). Bei den Whetstones ist er mehr als ein Drittel schneller als ein Compaq 386/25 mit 80387-Koprozessor (2400 KiloWs), muß sich aber dem gleichem Rechner mit Weitek-Koprozessor geschlagen geben (5220 KiloWs). Da damit hauptsächlich die Real-Arithmetik getestet wird, ist diese Leistung nur für bestimmte Gruppen von Anwendungen von Bedeutung, zeigt aber, daß der Sockel für den Weitek-Koprozessor nicht ganz umsonst da ist. Bei den Dhrystones bringt es ein Compaq 386/25 auf 10617.

Der Praxistest

Wenn auch Whetstones und Dhrystones einen Hinweis auf die Geschwindigkeit eines Rechners geben, so sind sie für die Praxis nicht immer repräsentativ. Deshalb haben wir ein Anwendungsprogramm aus der Wasserwirtschaft für einen exemplarischen Praxistest herangezogen. Neben den oben erwähnten Einsatzgebieten eines 486ers dürfte diese Rechnerklasse auch für Simulationen in wissenschaftlich-technischen Anwendungen interessant sein. Die rechenintensiven numerischen Lösungsverfahren stellen hinsichtlich Durchsatz und Speicherkapazität gehobene Anforderungen an die Technik. Um ein Programm aus dieser Kategorie handelte es sich bei diesem Test. Das Programm beeindruckte schon durch eine Größe der EXE-Datei von über 400 KByte. Es ist in traditioneller Weise in Fortran-77 programmiert und setzt den Einsatz eines Koprozessors voraus. Um nun eine Aussage hinsichtlich der Abarbeitungsgeschwindigkeit zu ermöglichen, sind in Tafel 2 die Rechenzeiten auf verschiedenen PCs und auf einem zur VAX 11/780-kompatiblen Rechner (robotron K 1840) der Rechenzeit am LGI 486 gegenübergestellt. Zwar soll der 486er etwa dreimal so schnell sein wie ein 386er, jedoch konnte dies im Test gegenüber dem Peacock 386-Tower nicht bestätigt werden. Hier wird deutlich, daß ein Hochleistungsprozessor nicht alles ist, sondern daß der PC als Gesamtgerät betrachtet werden muß, denn nur so kommt er in der Praxis zum Einsatz. Nichtsdestotrotz stellt der 486er eine Leistungsklasse dar, die im wahrsten Sinne des Wortes mit einem Rechenzentrum auf dem Schreibtisch verglichen werden kann. Im konkreten Einsatzfall läßt dieser Rechner ein problemloses Rechnen von Varianten mit dem besagten Simulationsprogramm zu, so daß sich der zu simulierende Prozeß gezielt

Dhrystone

Von Reinhold Weicker ursprünglich für den Unix-Bereich entwickelter synthetischer Benchmark, der ausschließlich die Leistung der CPU messen soll, allerdings sprachabhängig ist. Mit dem Ziel der Portierung wurde das Testprogramm zuerst in Ada, später in Pascal und C geschrieben. Es enthält Integer-, String- und Pointerverarbeitung sowie Feldzugriffe, aber keine Gleitkommabefehle und keine Betriebssystemrufe. Im einzelnen besteht es aus 53 % Zuweisungen, 33 % Kontrollanweisungen (Schleifen, Verzweigungen) und 15 % Unterprogrammrufen.

Whetstone

Das Testprogramm erhielt seinen Namen nach dem anfangs dafür verwendeten Whetstone-Compiler und dient insbesondere zur Untersuchung der Gleitkommaarithmetik des Prozessors und gegebenenfalls des Koprozessors. Es enthält einfach Gleitkommaoperationen, Winkelfunktionen, bedingte Sprünge, Funktionsaufrufe und Feldindizierungen. Der Whetstone-Benchmark repräsentiert damit wissenschaftliche und andere rechenintensive Anwendungen. Die Angabe erfolgt in Kilo- oder Mega-Whetstones.

untersuchen läßt. Erst wenn diese Arbeitsweise möglich ist, kann der Computer zu einem exklusiven Verständnis für die zu simulierenden Naturprozesse beitragen. Damit dürfte der 486er auch im ingenieurtechnischen Anwendungsbereich seine Interessenten finden. Nicht unerwähnt soll jedoch bleiben, daß diese enorme Leistung – so verlockend sie auch ist – ihren Preis hat, und der drückt sich nicht nur in Mark und Pfennig aus, sondern auch in einer zunehmenden Komplexität von Hard- und Software, die gehobene Anforderungen an den Anwender stellen. Es bleibt dabei nur zu hoffen, daß diese Systeme in den Händen der Anwender auch die Auslastung erfahren, für die sie gebaut wurden.

Fortsetzung von Seite 6

In der Zukunft werden die RISCs neben den CISCs eine der Hauptrichtungen in der Prozessorarchitektur sein, wobei auch die Symbiose beider Architekturlinien zur weiteren Leistungssteigerung genutzt werden wird.

Literatur

/1/ Vack, G.-U.: Mikrocontroller und Mikrorechner. Mikroprozessortechnik, Berlin 4 (1990) 5, S. 130
/2/ Bujara, A.: Ahrens, R.; Kucharzyk, U.: Analyse und Bewertung von 32-bit-Mikroprozessoren für den Einsatz unter Echtzeitbedingungen. Studie des Instituts für Informatik und Rechentechnik der Akademie der Wissenschaften, 1989
/3/ Kane, G.: MIPS R2000 RISC Architecture. Prentice Hall, Englewood Cliffs, 1987
/4/ Intel i860 – Wegbereiter einer neuen Leistungsklasse bei Mikroprozessoren. Mikroprozessortechnik, Berlin 3 (1989) 8, S. 251
/5/ SunNews, Firmenmitteilungen der Sun Microsystems GmbH, Mai 1990
/6/ Vack, G.-U.: Der Cache oder Wozu braucht eine CPU einen versteckten Speicher? Mikroprozessortechnik, Berlin 4 (1990) 4, S. 99

⊠ KONTAKT ☏

Akademie der Wissenschaften, Zentralinstitut für Kybernetik und Informationsprozesse, Abteilung 03, Rudower Chaussee 5, O-1199 Berlin; Tel. 6 74 26 34

Unix-Tip für Einsteiger

Mit der C-Shell doppelseitig drucken

Bernd Matzke, Delitzsch

Nutzer des zu Unix kompatiblen Betriebssystems Wega werden mit der Tatsache konfrontiert, daß viele Dokumentationen nur als Dateien auf Diskette vorliegen, also durch den Nutzer selbst ausgedruckt werden müssen. Da diese Dokumentationen einen erheblichen Umfang besitzen, ist ein doppelseitiges Bedrucken des Papiers wünschenswert. Diese Aufgabe übernimmt die im folgenden beschriebene Prozedurdatei für die Unix-Nutzeroberfläche C-Shell. Über den praktischen Nutzen hinaus kann die Analyse des C-Shell-Skriptes den Einstieg in die um interaktive Möglichkeiten erweiterte Nutzeroberfläche erleichtern.

Das Funktionsprinzip

Wega-Dokumentationen sind in einem Standardformat von 66 Zeilen je Seite formatiert. Aus diesem Grund ist der verwendete Drucker auf diese Seitenlänge einzustellen (Zeilenabstand 1/6 Zoll, Sprung über die Perforation ein, ASCII-Zeichensatz ohne Umlaute).
Das Kommando *split* zerlegt die zu druckende Datei in jeweils 66 Zeilen lange Teildateien. Diese Dateien erhalten standardmäßig den Namen xaa, xab ... xzz. Die Datei xaa enthält die erste Seite der Dokumentation, die Datei xab die zweite usw. Eine Ausnahme bildet lediglich die Datei xzz, die den gesamten Rest der aufgesplitteten Datei aufnimmt, falls die Dateien bis xzy dazu nicht ausreichen. Nun müssen diese Dateien nur noch in der richtigen Reihenfolge gedruckt werden, zuerst also die Dateien xaa, xac, xae ... auf die eine und danach die Dateien xab, xad, xaf ... auf die andere Seite des Papiers.
Damit auch Einzelblätter und Leporello-Reste verarbeitet werden können, kann in der Kommandozeile angegeben werden, nach wieviel Druckseiten das Papier gewendet bzw. gewechselt werden soll.
Beim Abbruch des Drucks wird der noch zu druckende Rest der Datei in eine Datei <name>.**r** gespeichert, die beim nächsten Start automatisch anstelle der ursprünglichen Datei verwendet wird.

Innerhalb der Prozedurdatei werden einige wichtige Möglichkeiten bei der C-Shell-Programmierung demonstriert. Hier sollen vor allem die kurz beschrieben werden, die aus herkömmlichen Programmiersprachen weniger oder gar nicht bekannt sind.

Die C-Shell ermöglicht eine interpretative Arbeit, daß heißt, bestimmte Elemente des Quelltextes können zur Ausführungszeit im Dialog durch aktuelle ersetzt werden. Diese Verfahrensweise ähnelt der Makro-Ersetzung in dBase, ist hier aber wesentlich perfekter gestaltet. Ausgelöst wird dieser Mechanismus durch das Dollarzeichen $. Der Ausdruck $<variable> wird bei der Abarbeitung des Programms durch den aktuellen Inhalt der Variablen ersetzt.
Die Zuweisung eines Ausdruckes zu einer Variablen erfolgt durch
set <variable> = <ausdruck>
Dabei ist zu beachten, daß Ausdrücke und damit Variableninhalte grundsätzlich Zeichen oder Zeichenketten sind. Sollen echte mathematische Operationen ausgeführt werden, so geschieht dies durch einen speziellen Operator.
Geschweifte Klammern begrenzen dabei den verwendeten Variablennamen. Unter der Annahme, das **argv[1]** die Zeichenkette *pas.dok* enthält, ist ${**argv[1]**}.**r** gleichbedeutend mit *pas.dok.r* (Zeile 11, 66).
Variablen können auch mehrere, durch Leerzeichen getrennte Elemente enthalten. In diesem Fall erfolgt der Zugriff auf das n-te Element mit $**var[n]**. Die Gesamtzahl der Elemente wird durch $#**var** festgestellt.
Es existiert darüber hinaus eine vordefinierte Variable argv, die die Parameter der Kommandozeile enthält. In Vergleichsausdrücken können neben den üblichen Vergleichsoperatoren = =, >, < und **!=** auch Dateioperationen zur Auswertung herangezogen werden. So prüft beispielsweise die Funktion (**−e** <name>), ob die angegebene Datei existiert, die Funktion (**−z** <name>) dagegen testet, ob die Datei leer ist. Das Kommando *echo* übergibt die nachfolgende, meist nicht extra gekennzeichnete Zeichenkette an das Standardausgabeausgabegerät; im allgemeinen an den Bildschirm. Mit der Option **−n** wird der Zeilenwechsel nach Ausgabe der Zeichenkette unterdrückt.
Da in der C-Shell aber alle ASCII-Sonderzeichen eine spezielle Bedeutung besitzen, ist es zweckmäßig, die durch das Kommando *echo* auszugebende Zeichenkette in Anführungszeichen (' oder ") einzuschließen. Der Variablen **a** werden jetzt alle Dateinamen zugewiesen, die dem Muster x?? entsprechen, also alle Dateinamen, die mit x beginnen und insgesamt drei Buchstaben lang sind (Zeile 29).
In der nun folgenden Schleife wird unter anderem die Anzahl der Elemente in der Variablen a getestet. Die Schleife wird so lange durchlaufen, bis a kein Element mehr enthält. Innerhalb der Schleife werden zunächst 3 Dateien angelegt, in dem die Ausgabe von **cat/dev/null** in diese drei Dateien umgeleitet wird. Dabei ist /dev/null sozusagen der *Papierkorb* des Systems. Alles, was in diese *Datei* geschrieben wird, ist auf Nimmerwiedersehen verloren; Ausgaben aus dieser Datei – wie im Skript – enthalten absolut nichts. Die so erzeugten Dateien *us*, *gs* und *rmdat* sind also mit Sicherheit leer. Die Zeilen 36 bis 47 sind das eigentliche Kernstück des Skriptes.
Es wird eine zweite Schleife eröffnet, die so oft durchlaufen wird, bis die gewünschte Anzahl von Seiten bearbeitet wurde. Wenn in a noch ein Element vorhanden ist, so wird die

```
 1  # C-Shell-Skript DB zum doppelseitigen Druck von Texten
 2  onintr interrupt
 3  echo
 4  if ($#argv != 2) then
 5    echo
 6    echo "Aufruf: db <filename> <blattzahl>"
 7    echo
 8    exit
 9  endif
10  if (-e ${argv[1]}.r) then
11    set filename = ${argv[1]}.r
12  else
13    if (-e $argv[1]) then
14      set filename = $argv[1]
15    else
16      echo
17      echo $argv[1] existiert nicht!
18      echo
19      exit
20    endif
21  endif
22  if (-z $filename) then
23    echo
24    echo $filename ist leer!
25    echo
26    exit
27  endif
28  split -66 $filename
29  set a = (x??)
30  set weiter = j
31  while (($weiter != n) && ($weiter != N) && ($#a > 0))
32    set i = 1
33    cat /dev/null > us
34    cat /dev/null > gs
35    cat /dev/null > rmdat
36    while ($i <= $argv[2])
37      if ($#a > 0) then
38        cat $a[1] >> us
39        echo $a[1] >> rmdat
40        shift a
41      endif
42      if ($#a > 0) then
43        cat $a[1] >> gs
44        echo $a[1] >> rmdat
45        shift a
46      endif
47      @ i++
48    end
49    echo
50    echo "Papier einspannen, weiter mit <ET>"
51    gets
52    cat us > /dev/lp
53    echo
54    echo "Papier wenden, weiter mit <ET>"
55    gets
56    cat gs > /dev/lp
57    echo
58    echo -n "Weitermachen (j/n):"
59    set weiter = `gets`
60    foreach b (`cat rmdat`)
61      rm $b
62    end
63  end
64  interrupt:
65  echo
66  cat x?? > ${argv[1]}.r
67  rm x??
68  if (-z ${argv[1]}.r) then
69    echo Datei vollständig gedruckt
70    rm ${argv[1]}.r
71  endif
72  rm us
73  rm gs
74  rm rmdat
```

***Bild 1** Die C-Shell-Prozedur db bedruckt Papier doppelseitig.*

erste in a enthaltene Datei mittels *cat* ausgegeben, die Ausgabe aber in die Datei *us*, in der die ungeraden Seiten gespeichert werden, umgeleitet. Dabei ist zu beachten, daß die Umleitung mit dem Anfügeoperator >> erfolgen muß. Der Dateiname selbst wird an die Datei *rmdat* angefügt, in der die Namen der zu löschenden Dateien zwischengespeichert werden. Anschließend verschiebt *shift* den Inhalt von a um ein Element nach links. Dabei geht das erste Element verloren, das zweite tritt an die Stelle des ersten usw. Der Vorgang wird nun mit der ersten Datei in a, die jetzt aber eine gerade Seite enthält, wiederholt, wobei an die Datei *gs* angefügt wird. Nach dem Durchlauf wird der Schleifenzähler i um eins erhöht. Da es sich hierbei um eine mathematische Operation handelt, muß dies durch den Operator @ angezeigt werden. Verwendet wird hier zur Erhöhung von i der Inkrement-Operator ++, der der Programmiersprache C entlehnt ist. Im Anschluß an das Umspeichern werden die Dateien *us* und *gs* ausgedruckt. Erst, wenn das Ausdrucken der beiden Dateien erfolgreich abgeschlossen wurde, werden die entsprechenden x??-Dateien gelöscht, deren Namen in der Datei *rmdat* gespeichert wurden.

Hierzu wird ein weiterer Mechanismus der C-Shell benutzt. Wird ein Kommando in Accent graves (Zeichen ') eingeschlossen, wird es abgearbeitet und seine Ausgabe an dieser Stelle in das Programm eingefügt. Die Angabe **'cat rmdat'** bewirkt also, daß der Inhalt von rmdat ausgegeben wird und die in dieser Datei gespeicherten Dateinamen nacheinander der Laufvariablen b der Foreach-Anweisung zugewiesen werden. In der Zeile 59 wird dieser Mechanismus ebenfalls benutzt. Das Kommando *gets* liest eine Zeichenkette vom Terminal und gibt diese wieder auf das Terminal aus. Demzufolge bewirkt

set weiter = 'gets'

daß der Variablen *weiter* die Ausgabe von *gets* zugewiesen wird, also die am Terminal eingegebene Zeichenkette. Das Kommando *gets* liest eine Zeichenkette bis zum ersten Trennzeichen (Leerzeichen, Tabulator). Die äußere While-Schleife testet aber auf Übereinstimmung mit einem einzelnen Zeichen. Wird anstelle eines einzelnen Zeichens n eine Zeichenkette (z. B. nn) eingegeben, so wird dies nicht als Abbruchbedingung für die Schleife gewertet.

Nach Beendigung dieser Schleife werden alle noch existierenden, also noch nicht ausgedruckten, x??-Dateien in die Datei *<name>.r* geschrieben.

Abschließend möchte ich noch die Anweisung *onintr* erläutern. Diese Anweisung definiert eine Sprungmarke, zu der bei einem Interrupt (in der Regel ein Programmabbruch durch die Del-Taste) gesprungen wird.

Da der Programmabbruch auch während des Drucks erfolgen kann, dürfen in diesem Fall die in *rmdat* gespeicherten Dateien nicht gelöscht werden. Der Programmabschluß – insbesondere das Retten der noch nicht gedruckten Seiten in der Datei <name>.r – kann aber wie bei normaler Programmbeendigung erfolgen.

Sollen die für die Interrupt-Bearbeitung vorgesehenen Anweisungen bei normalem Programmablauf nicht abgearbeitet werden, so muß vor der Interrupt-Marke das Programmende (Exit-Anweisung) programmiert werden.

Literatur

/1/ Wega-Programmierhandbuch, Teil 1/1 und 1/2. VEB EAW Treptow, Berlin 1987

/2/ Wega-Dienstprogramme, Teil D: C-Shell, VEB EAW Treptow, Berlin 1988

Grundlagen des logischen Programmierens Teil 2

Dr. Konrad Schultz, Chemnitz

Im zweiten Teil geht es um die Darstellung strukturierten Wissens in der Wissensbasis eines logischen Programms.

Konjunktionen

Die Aussage „Hans liebt Grete, und Grete liebt ihre Mutter." ist eine zusammengesetzte Aussage, keine einfache, und wird in der mathematischen Logik geschrieben als:

liebt(hans,grete) ∧ liebt(grete,mutter_von(grete))

Diese Konjunktion der beiden Aussagen liebt(hans,grete) und liebt(grete,mutter_von(grete)) ist genau dann wahr, wenn ihre beiden Bestandteile wahr sind. Eine Konjunktion ist falsch, wenn wenigstens ein Bestandteil falsch ist. In Prolog wird statt ∧ ein Komma geschrieben. Allerdings ist die Formulierung

liebt(hans,grete),liebt(grete,mutter_von(grete)).

nicht sinnvoll. Die beiden Fakten

liebt(hans,grete).
liebt(grete,mutter_von(grete)).

lösen die beabsichtigte Wirkung aus: Beide Teilaussagen werden vom Computer als wahr angesehen. Konjunktionen spielen in Prolog eine Rolle als Bestandteile von Regeln und Fragen, zu denen wir noch kommen werden.

Wir sehen übrigens, daß das Komma in Prolog zwei verschiedene Bedeutungen hat: Es trennt Argumente, und es verbindet zwei Aussagen konjunktiv.

Alternativen

Die Aussage „Es schneit, oder es ist kalt" ist eine zusammengesetzte Aussage und wird in der mathematischen Logik geschrieben als

schneit ∨ kalt

Diese Alternative (auch Disjunktion genannt) der Aussagen *schneit* und *kalt* ist genau dann wahr, wenn wenigstens einer der Bestandteile wahr ist. Für Konjunktion und Alternative können wir also die folgenden Wertetafeln aufstellen

∧	wahr	falsch
wahr	wahr	falsch
falsch	falsch	falsch

∨	wahr	falsch
wahr	wahr	wahr
falsch	wahr	falsch

In Prolog wird **;** statt ∨ benutzt, allerdings werden Alternativen dort nur selten explizit geschrieben. Eine Eingabe der Alternative

schneit;kalt

ist nicht möglich, und das hängt, wie schon bei der Partikularisierung, mit dem konstruktiven Charakter von Prolog zusammen. Konstruktiv gesehen, ist die Wahrheit einer Alternative erst dann gesichert, wenn einer der Bestandteile als wahr nachgewiesen wurde. Der Computer wird also *schneit;kalt* erst dann als wahr ansehen, wenn er über einen der Fakten

schneit.

oder

kalt.

verfügt (durch Eingabe oder abgeleitet auf Grund entsprechender Regeln).

Aussagenlogische Äquivalenz, konjunktive Normalformen

φ_1 und φ_2 seien Aussagen. φ_1 und φ_2 sind genau dann aussagenlogisch äquivalent, wenn sie entweder beide wahr oder beide falsch sind, und zwar unabhängig davon, ob ihre Bestandteile wahr oder falsch sind. Wir schreiben dann $\varphi_1 \equiv \varphi_2$, erweitern die Definition von ≡ aber später zur logischen Äquivalenz. So gilt

schneit ≡ schneit,
schneit ≡ schneit ∧ schneit,
schneit ≡ schneit ∨ schneit,
schneit ∧ kalt ≡ kalt ∧ schneit,
kalt ∨ schneit ≡ schneit ∨ kalt,
(schneit ∧ kalt) ∧ stark(wind) ≡
≡ schneit ∧ (kalt ∧ stark(wind)),
(schneit ∨ kalt) ∨ stark(wind) ≡
≡ schneit ∨ (kalt ∨ stark(wind)),

denn unabhängig davon, ob es schneit, kalt ist, stark weht oder nicht, steht auf beiden Seiten von ≡ eine wahre oder auf beiden Seiten von ≡ eine falsche Aussage. Wichtig sind auch die Distributivgesetze. Wenn $\varphi_1, \varphi_2, \varphi_3$ Aussagen sind, so gilt

$$\varphi_1 \wedge (\varphi_2 \vee \varphi_3) \equiv (\varphi_1 \wedge \varphi_2) \vee (\varphi_1 \wedge \varphi_3)$$

und

$$\varphi_1 \vee (\varphi_2 \wedge \varphi_3) \equiv (\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3)$$

(Zum Vergleich: Beim Zahlenrechnen gilt nur das Distributivgesetz $a * (b + c) = (a * b) + (a * c)$, aber das andere ist falsch, denn im allgemeinen gilt nicht

$$a + (b * c) = (a + b) * (a + c).$$

Durch wiederholte Anwendung des zweiten Distributivgesetzes von links nach rechts läßt sich jede mit Hilfe von ∧ und ∨ zusammengesetzte Aussage aussagenlogisch äquivalent in der Form $\varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n$ darstellen, wobei alle φ_i einfache Aussagen oder Alternativen von einfachen Aussagen sind. In diesem Falle haben wir eine Aussage in konjunktiver Normalform vor uns, bei wiederholter Anwendung des ersten Distributivgesetzes kommen wir entsprechend zu einer alternativen (oder disjunktiven) Normalform.

So ist zum Beispiel $(\varphi_1 \wedge \varphi_2) \vee (\varphi_3 \vee \varphi_4) \vee (\varphi_5 \wedge \varphi_6)$ bereits in einer alternativen Normalform. Durch wiederholte Anwendung des zweiten Distributivgesetzes kommen wir zu einer konjunktiven Normalform

$$(\varphi_1 \vee \varphi_3 \vee \varphi_5) \wedge (\varphi_1 \vee \varphi_3 \vee \varphi_6) \wedge$$
$$\wedge (\varphi_1 \vee \varphi_4 \vee \varphi_5) \wedge (\varphi_1 \vee \varphi_4 \vee \varphi_6) \wedge$$
$$\wedge (\varphi_2 \vee \varphi_3 \vee \varphi_5) \wedge (\varphi_2 \vee \varphi_3 \vee \varphi_6)$$
$$\wedge (\varphi_2 \vee \varphi_4 \vee \varphi_5) \wedge (\varphi_2 \vee \varphi_4 \vee \varphi_6)$$

An diesem Beispiel deutet sich an, daß die Herstellung einer Normalform von der Methode her sehr einfach ist, aber bei längeren

Aussagen praktisch aufwendig wird. Um eine Alternative aus 10 Konjunktionen zu je 2 Gliedern (eine alternative Normalform) in eine konjunktive Normalform umzuformen, muß man eine Konjunktion von $2^{10} = 1024$ Alternativen zu je 10 Gliedern bilden. Diese exponentielle Komplexität ist für die logische Programmierung ein ernstes Hindernis. Theoretisch freilich ist die Herstellung der Normalform durch den Computer immer möglich.

Implikationen und Regeln

Betrachten wir die Aussage „Wenn Hans Grete sieht, so ist er glücklich." Das ist wiederum eine aus den einfachen Aussagen „Hans sieht Grete" und „Hans ist glücklich" zusammengesetzte Aussage, die in der mathematischen Logik Implikation genannt und geschrieben wird als

sieht(hans,grete) → glücklich(hans)

oder seltener als

glücklich(hans) ← sieht(hans,grete)

Und fast in derselben Form kann die Implikation, jetzt **Regel** genannt, auch dem Prolog-Computer eingegeben werden:

glueckIich(hans):–sieht(hans,grete).

Beachten Sie hierbei den Punkt, der Fakten und Regeln – gemeinsam **Klauseln** genannt – abschließt (und vor allem trennt). Die Implikation $\varphi_1 \rightarrow \varphi_2$ gilt als falsch, wenn die **Voraussetzung** φ_1 wahr ist, aber die **Behauptung** φ_2 falsch ist; in allen anderen Fällen ist die Implikation wahr. Wir erhalten somit:

→	wahr	falsch
wahr	wahr	falsch
falsch	wahr	wahr

So wie wir $\varphi_1 \rightarrow \varphi_2$ bzw. φ_2:–φ_1. logisch lesen als

Wenn φ_1 (wahr ist), so (ist) auch φ_2 (wahr).

so bedeutet das für den Computer

Um φ_2 wahr zu machen, genügt es, φ_1 wahr zu machen.

oder

Um φ_2 zu realisieren, genügt es, φ_1 zu realisieren.

oder

Der Versuch, φ_2 zu realisieren (der **Aufruf** von φ_2), zieht den Versuch nach sich, φ_1 zu realisieren (zieht den Aufruf von φ_1 nach sich).

Das bedeutet eine inhaltliche Erweiterung, denn die Realisierung gewisser Atome durch den Computer besteht nicht immer (nur) darin, sie als wahr zu erkennen (logische oder deklarative Interpretation), sondern bedeutet zuweilen (auch) Handlungen des Computers (prozedurale Interpretation). So wird etwa *write(hans)* realisiert, indem das Wort *hans* ausgegeben wird; der Aufruf von *write(hans)* ist immer erfolgreich, hat aber mit logischer Wahrheit wenig zu tun.

Die Logikprogrammierung, insbesondere Prolog, lebt davon, daß verschiedene Interpretationen möglich sind, die deklarative zur Problembeschreibung und die prozedurale zur Steuerung insbesondere der Peripherie, aber auch des Computers selbst. Wenn sogenannte Seiteneffekte wie Druckausgaben im Spiel sind, ist die prozendurale Deutung die bestimmende. Die beiden Seiten von Prolog-Regeln unterliegen gewissen Einschränkungen, auf die wir zu sprechen kommen, nachdem wir einige aussagenlogische Äquivalenzen studiert haben. Es gilt

$$\varphi_1 \rightarrow (\varphi_2 \wedge \varphi_3) \equiv (\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_1 \rightarrow \varphi_3)$$
$$(\varphi_1 \vee \varphi_2) \rightarrow \varphi_3 \equiv (\varphi_1 \rightarrow \varphi_3) \wedge (\varphi_2 \rightarrow \varphi_3)$$
$$\varphi_1 \rightarrow (\varphi_2 \rightarrow \varphi_3) \equiv (\varphi_1 \wedge \varphi_2) \rightarrow \varphi_3$$
$$(\varphi_1 \rightarrow \varphi_2) \rightarrow \varphi_3 \equiv (\varphi_2 \rightarrow \varphi_3) \wedge (\varphi_1 \vee \varphi_3)$$
$$(\varphi_1 \vee \varphi_2) \vee (\varphi_3 \rightarrow \varphi_4) \equiv$$
$$\equiv (\varphi_1 \wedge \varphi_3) \rightarrow (\varphi_2 \vee \varphi_4)$$

Wenn wir mit Hilfe von $\wedge$, $\vee$ und $\rightarrow$ zusammengesetzte Aussagen auf das zweite Distributivgesetz und die eben angegebenen Äpuivalenzen von links nach rechts so oft wie möglich anwenden, kommen wir zur konjunktiven Normalform in Klauselform. Eine konjunktive Normalform in Klauselform ist eine Konjunktion $\varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n$, wobei jedes φ_i entweder eine Alternative von einfachen Aussagen ist oder die Gestalt $(\psi_1 \wedge \ldots \wedge \psi_m) \rightarrow (\psi_{m+1} \vee \ldots \vee \psi_k)$ mit einfachen Aussagen $\psi_1, \ldots, \psi_k$ hat. Im Falle von $k = m + 1$ sprechen wir von Horn-Klauseln, den Fall von zugleich $m = 0$ (einfache Aussage ψ_1) dabei zuweilen mit einbezogen.

In Prolog dürfen nun Implikationen (hier Regeln genannt) *nur* in der Form von (meist generalisierten, siehe unten) Horn-Klauseln eingegeben werden, also in der Form

ψ_0:–ψ_1, $\psi_2 \ldots \psi_m$.

mit abschließendem Punkt. Die Bedeutung (die Wirkung auf den Computer) ist die:

Der Aufruf von ψ_0 zieht die Aufrufe von ψ_1, $\psi_2, \ldots, \psi_m$ in dieser Reihenfolge nach sich.

Zum Beispiel bedeutet

schreibe(max,meyer):–write('Max '),write('Meyer').

daß der erfolgreiche Aufruf von *schreibe(max,meyer)* nach dieser Regel die Ausgabe von *Max Meyer* nach sich zieht. Wir sehen hier auch, daß die prozedurale Interpretation Momente ins Spiel bringt, die der Logik fremd sind. Während nämlich

$$(\psi_1, \wedge \psi_2) \rightarrow \psi_0 \equiv (\psi_2 \wedge \psi_1) \rightarrow \psi_0$$

gilt, kommt es doch auf die Reihenfolge der *write*-Aufrufe in der *schreibe*-Regel an. Die Aufrufe in Prolog gehen konsequent von links nach rechts vor sich. (Wenn in Prolog-Erweiterungen die Aufrufe parallel erledigt werden sollen, wird das in der Regelschreibweise vermerkt.)

Warum werden in Prolog nur Horn-Klauseln verwendet? Wir haben schon bemerkt, warum in Prolog die Eingabe von Alternativen nicht möglich ist. Eine Alternative

$\psi_0;\psi_1$.

könnte gleichwertig durch die Klauseln

verum.

$\psi_0;\psi_1$:–verum.

ersetzt werden. Wir müssen also ; links von :– verbieten, um den konstruktiven Charakter von Prolog zu erhalten. ; rechts von :– ist definierbar, denn aus den angegebenen Äquivalenzen folgt zum Beispiel, daß

winterwetter:–schneit;kalt.

gleichwertig durch

winterwetter:–schneit.

winterwetter:–kalt.

ersetzt werden kann.

Ein anderer Grund für die Beschränkung auf Horn-Klauseln ist der, daß der Computer mit einer so eingeschränkten Logikprogrammierung wesentlich effektiver arbeiten kann, es lassen sich leichter schnellere Beweisalgorithmen implementieren. Meist wird nur diese Begründung angegeben.

Regeln mit Variablen

Wir haben bisher keine Variablen in Regeln betrachtet. Es können aber sehr wohl Variablen vorkommen. Wir gehen etwa von der Regel

glueckIich(hans):–liebt(hans,grete),sieht(hans,grete).

aus. Zu dieser Regel – logisch einer zusammengesetzten Aussage – können wir eine Generalisierung bilden

glueckIich(Mann):–liebt(Mann,Frau),sieht(Mann,Frau).

mit der Bedeutung

Für jeden Mann und jede Frau gilt: Wenn der Mann die Frau liebt, so ist er glücklich, wenn er sie sieht.

Mann und *Frau* sind Variablen (mit groß geschriebenen Anfangsbuchstaben).

In Prolog sind generalisierte Regeln (Horn-Klauseln) – ebenfalls als Regeln (Horn-Klauseln) bezeichnet – zugelassen. Wesentlich ist aber, daß die Regel als Ganzes generalisiert ist, also als

∀X∀Y (liebt(X,Y) ∧ sieht (X,Y) → glücklich(X))

zu lesen ist und nicht etwa als

∀X∀Y liebt(X,Y) ∧ ∀X∀Y sieht(X,Y) → ∀X glücklich(X)

eine Zusammensetzung von generalisierten Aussagen. Die eben genannte zusammengesetzte Aussage ist keine Horn-Klausel, denn auf den beiden Seiten von → dürfen in Horn-Klauseln nur einfache Aussageformen (keine Generalisierungen) rechts und Konjunktionen von einfachen Aussagenformen links vorkommen. Man beachte also: Wenn die gleiche Variable in einer Regel, wie überhaupt einer Klausel, mehrfach auftritt, steht sie jedes Mal (generalisierend) für dasselbe Objekt. Wie aus generalisierten und partikularisierten Aussagen zusammengesetzte Aussagen gleichwertig umgeformt werden können, erfahren Sie im 3. Teil dieser Artikelfolge.

Prolog ist (abgesehen von *einer* abweichenden Implementation) eine typfreie Sprache. Alle Variablen haben als zugelassene Werte beliebige Prolog-Objekte. Eine Reservierung von Variablen beispielsweise für ganze Zahlen, wie in anderen Programmiersprachen, erfolgt nicht.

Die Negation

Bei der Bildung zusammengesetzter Aussagen fehlt uns noch die Negation. „Die Zitrone ist nicht gelb", in der mathematischen Logik geschrieben ¬*gelb(zitrone)*, ist die Negation der Aussage *gelb(zitrone)*. Die Aussage $\neg\varphi$ ist wahr, falls φ nicht wahr ist, und umgekehrt.

In Prolog ist die Formulierung von Negationen nicht möglich. Wir sehen selbst ja auch nur, daß die Zitrone grün ist, und schließen daraus erst, daß sie nicht als gelb anzusehen ist. Der Prolog-Interpreter geht ähnlich vor, er sieht *gelb(zitrone)* nur dann als wahr an, wenn das aus seiner Wissensbasis zwingend (logisch) folgt; wenn das nicht sein kann, wird sie – allerdings unabhängig von der Wahrheit von *gruen(zitrone)* – als *nicht-gelb* angesehen (siehe Teil 3). Bei Zulassung von Negativ-Eingaben könnten wir auch die Alternative eingeben wegen

$$\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2)$$

und ähnlich auch Partikularisierungen, der konstruktive Charakter von Prolog wäre verloren.

Die Negation steht der Herstellung von Normalformen nicht im Wege. Es gilt nämlich

$$\neg\neg\varphi \equiv \varphi$$
$$\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$$
$$\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$$
$$\neg(\varphi \rightarrow \psi) \equiv \varphi \vee \neg\psi$$

Im nächsten Abschnitt wird die Definition der Äquivalenz $\equiv$ so erweitert, daß dann auch gilt

$\neg\forall X\,\varphi \equiv \exists X\,\neg\varphi$
$\neg\exists X\,\varphi \equiv \forall X\,\neg\varphi$
So können wir durch wiederholte Anwendungen dieser Gesetze alle Negationszeichen soweit nach innen treiben, daß sie nur noch vor einfachen Aussagenformen stehen. Eine verneinte oder unverneinte Aussagenform heißt in der Logikprogrammierung **Literal**. Anschließend lassen sich beliebige der bisher betrachteten Normalformen bilden, die Literale werden dabei genauso behandelt wie einfache Aussagenformen oder bisher einfache Aussagen. Bei konjunktiven Normalformen in Klauselform lassen sich wegen
$(\varphi_1 \wedge \neg\varphi_2) \rightarrow (\varphi_3 \vee \neg\varphi_4) \equiv (\varphi_1 \wedge \varphi_4) \rightarrow (\varphi_2 \vee \varphi_3)$
einige Negationen einsparen. Dies wird aber keine weitere Rolle spielen, da wir uns auf den theoretischen Hintergrund von Prolog konzentrieren und in Prolog ja die Formulierung der Negation nicht möglich ist – nur der eventuelle Schluß auf die Abwesenheit der positiven Eigenschaft (Teil 3).

Fragen, das logische Folgern und verwandte Begriffe

Stellen Sie sich vor, Sie haben dem Computer allerhand Wissen in Form von Klauseln eingegeben, nicht unbedingt Horn-Klauseln. Nutzen können Sie eine solche Wissensbasis nur durch Fragen, in Prolog etwa notiert als

?-liebt(hans,eva).

Hier wird also im Prinzip eine einfache Aussage in Frageform eingegeben. Der Computer soll nun antworten, ob Hans Eva liebt. Dabei weiß er gar nicht, was Liebe inhaltlich ist, und von Hans und Eva weiß er außer dem Namen genau das, was in der Wissensbasis abgespeichert ist. Der Computer wird dennoch die Antwort

yes

genau dann ausgeben, wenn die Aussage *liebt(hans,eva)* aus der Wissenbasis logisch folgt. Und sie folgt aus der Wissensbasis logisch genau dann, wenn unabhängig davon, was für Objekte wir uns unter *hans* und *eva* vorstellen, unabhängig davon, für welche Objekte die Beziehung *liebt* zutrifft, unabhängig davon, wie die übrigen Konstanten in der Wissensbasis interpretiert werden, allein aus der Annahme, daß alle Klauseln der Wissensbasis wahr sind, folgt, daß auch die Aussage *liebt(hans,eva)* wahr ist.

Geben wir die genaue mathematische Definition: Φ sei eine Menge von Aussagen, φ eine Aussage. Eine **Interpretation** von $\Phi \cup \{\varphi\}$ ist gegeben durch

① einen gemeinsamen nichtleeren Individuenbereich für alle Variablen (die Betrachtung verschiedener Individuenbereiche für verschiedene Variablen ist unnötig und kompliziert die Definition)

② eine Deutung jeder Individuenkonstanten durch einen Gegenstand aus dem Individuenbereich

③ Eine Deutung jeder Funktionenkonstanten durch eine Funktion (gleicher Stellenzahl) über diesem Individuenbereich

④ eine Deutung jeder Prädikatenkonstanten durch ein Prädikat (gleicher Stellenzahl) über dem Individuenbereich, also eine Menge von k-Tupeln von Gegenständen aus dem Individuenbereich, für die das Prädikat (der Stellenzahl k) zutreffen soll, für die es wahr ist.

Wenn eine Interpretation von $\Phi \cup \{\varphi\}$ gegeben ist, so ist für jede Aussage aus $\Phi \cup \{\varphi\}$ bezüglich dieser Interpretation festgelegt, ob sie wahr ist oder nicht. So ist *kleiner(3,4)* wahr, falls der Individuenbereich die Gegenstände 3 und 4 umfaßt und die Prädikatenkonstante *kleiner* als die übliche <-Relation interpretiert wird. *kleiner(3,4)* ist aber falsch, wenn wir *kleiner* boshafterweise durch die >-Relation interpretieren, was nicht verboten ist. Wenn für jede einfache Aussage bekannt ist, ob sie bei einer Interpretation wahr ist oder nicht, so gilt das auch für zusammengesetzte Aussagen:

$\neg\varphi$ ist genau dann wahr, wenn φ nicht wahr ist.

$\varphi_1 \wedge \varphi_2$ ist genau dann wahr, wenn φ_1 und φ_2 beide wahr sind.

$\varphi_1 \vee \varphi_2$ ist genau dann wahr, wenn wenigstens eine der Aussagen φ_1 und φ_2 wahr ist.

$\varphi_1 \rightarrow \varphi_2$ ist genau dann wahr, wenn $\neg\varphi_1 \vee \varphi_2$ wahr ist.

$\forall X\varphi$(name/X) ist genau dann wahr, wenn die Aussage φ

unabhängig davon wahr wird, wie die Interpretation bezüglich der Individuenkonstanten *name* (nur hier) abgeändert wird. Die Aussagenform φ(name/X) entstehe dabei dadurch aus der Aussage φ, daß die Individuenkonstante *name* an allen Stellen durch die Variable X ersetzt wird.

$\exists X\varphi$(name/X) ist genau dann wahr, wenn die Interpretation bezüglich *name* (nur hier) so abgeändert werden kann, daß φ wahr wird. Eine Aussage der Form $\forall X\varphi$ bzw. $\exists X\varphi$ ist also genau dann wahr, wenn φ(X/gegenstand) für alle Gegenstände bzw. wenigstens einen Gegenstand *gegenstand* aus dem Individuenbereich wahr ist.

Eine Interpretation, die alle Aussagen von Φ (bzw. die Aussage φ) wahr macht, heißt **Modell** von Φ (bzw. φ). φ folgt logisch aus Φ (geschrieben $\Phi \models \varphi$) genau dann, wenn jede Interpretation von $\Phi \cup \{\varphi\}$, die Modell von Φ ist, auch ein Modell von φ ist.

φ und ψ sind logisch äquivalent (geschrieben $\varphi \equiv \psi$) genau dann, wenn φ aus $\{\psi\}$ folgt und ψ aus $\{\varphi\}$, wenn φ und ψ im wesentlichen also dieselben Modelle haben („im wesentlichen" heißt: sofern nur alle Konstanten aus φ und aus ψ interpretiert sind.)

φ ist logisch gültig oder allgemeingültig (geschrieben $\models \varphi$) genau dann, wenn jede Interpretation von φ auch ein Modell von φ ist (genau dann, wenn φ aus der leeren Menge $\emptyset$ folgt). Also: Der Computer gibt auf die Frage

?-liebt(hans,eva).

genau dann *yes* aus, wenn gilt:

Wissensbasis $\models$ liebt(hans,eva)

Wie soll aber der Computer die unendlich vielen Modelle mit zum Teil unendlich großen Individuenbereichen alle durchmustern? Dies wird durch eine Beschränkung der betrachteten Modelle möglich.

Eine Interpretation heißt genau dann **kanonisch**, wenn alle Grundterme durch sich selbst interpretiert werden (also als gewisse Zeichenketten) und als Individuenbereich gerade der Bereich der Grundterme gewählt wird. Es reicht nun aus, kanonische Interpretationen zu betrachten: Es gilt genau dann $\Phi \models \varphi$, wenn nur jede kanonische Interpretation von $\Phi \cup \{\varphi\}$, die ein Modell von Φ ist, auch ein Modell von φ ist.

Wir brauchen uns also nicht um beliebige Bedeutungen der Konstanten zu kümmern, sondern können die Grundterme generell durch sich selbst interpretieren, was für den Computer sicher das einfachste ist, und den Prädikaten nur vorzuschreiben, für welche Argumentlisten von Grundtermen sie zutreffen sollen. Es stellt sich bei der Computerbehandlung aber noch das Problem, daß es unter Umständen nur unendliche kanonische Modelle von $\Phi \cup \{\neg\varphi\}$ gibt, nur unendliche Modelle, die die Annahme von $\Phi \models \varphi$ widerlegen können. Sei etwa ψ gleich

$\forall X\,\neg r(X,X) \wedge \forall X\forall Y\forall Z(r(X,Z) \wedge r(Y,Z) \rightarrow r(X,Z)) \wedge \forall X\exists Y\,r(X,Y)$.

Es gibt Modelle von ψ (irreflexive unendliche Ordnungen wie < im Bereich der natürlichen Zahlen), aber jedes Modell von ψ ist unendlich. Somit ist jede endliche kanonische Interpretation von $\neg\psi$ ein Modell von $\neg\psi$, ohne daß gilt $\models\neg\psi$ oder $\emptyset\models\neg\psi$. Unendliche Modelle können vom Computer nicht als Ganzes untersucht werden. Es gibt aber mit Hilfe einiger logischer Umformungen Auswege.

Pränexe Normalformen

Für Aussagen ψ, $\forall X\psi$ und $\exists X\psi$ gilt:
$\neg\forall X\psi \equiv \exists X\neg\psi$
$\neg\exists X\psi \equiv \forall X\neg\psi$
$\varphi \wedge \forall X\psi \equiv \forall X(\varphi \wedge \psi)$
$\varphi \vee \forall X\psi \equiv \forall X(\varphi \vee \psi)$
$\varphi \wedge \exists X\psi \equiv \exists X(\varphi \wedge \psi)$
$\varphi \vee \exists X\psi \equiv \exists X(\varphi \vee \psi)$
$\varphi \rightarrow \forall X\psi \equiv \forall X(\varphi \rightarrow \psi)$
$\varphi \rightarrow \exists X\psi \equiv \exists X(\varphi \rightarrow \psi)$
$\forall X\psi \rightarrow \varphi \equiv \exists X(\psi \rightarrow \varphi)$
$\exists X\psi \rightarrow \varphi \equiv \forall X(\psi \rightarrow \varphi)$

so daß durch wiederholte Anwendungen dieser Äquivalenzen von links nach rechts jede Aussage logisch äquivalent in eine Form $Q_1X_1 \ldots Q_nX_n\psi$ überführt werden kann, wobei jedes Q_i gleich $\forall$ oder gleich $\exists$ ist und φ eine Aussagenform ohne $\forall$ und ohne $\exists$ ist, also nur noch mit Hilfe von $\neg$, $\wedge$, $\vee$ und $\rightarrow$ aus einfachen Aussagenformen zusammengesetzt ist. Eine solche Form einer Aussage heißt pränexe Normalform. (Es sei bemerkt, daß man äquivalente kontrapränexe Normalformen, wo sich $\forall$ und $\exists$ jeweils nur auf einfache Aussagenformen beziehen, im allgemeinen nicht bilden kann.).

In Prolog sind *nur* pränexe Normalformen zugelassen, sogar nur Skolemsche Normalformen, auf die wir im Teil 3 zu sprechen kommen.

wird fortgesetzt

✉ KONTAKT ☎

Technische Universität Chemnitz, Sektion Informatik, Postfach 964, O-9010 Chemnitz; Tel. 66 82 52

„Stundenlang könnte ich Ihnen beim Programmieren zusehen!"

Zeichnung: Steger

Connections

Teil 1

Alle wichtigen Rechner-Anschlüsse auf einen Blick

Andreas Stiller, Carsten Meyer

Wer kennt sie nicht, die Fragen: Wie war doch noch die Pin-Belegung der seriellen „Schnitte" oder der Grafik-Karte, ist die Buchse am Rechner xy weiblich oder männlich, 25- oder 9polig oder gar in Mini-DIN-Ausführung, wo ist +5 Volt bei der Floppy-Spannungsversorgung et cetera? Höchste Zeit, alle wesentlichen Stecker- und Buchsenausführungen sowie Bepinnungen mal übersichtlich zusammenzustellen, damit Sie die richtigen Kontakte knüpfen können.

Drucker

Bei der Druckerschnittstelle findet man inzwischen fast nur noch den „Quasi-Standard", welcher nach der amerikanischen Firma Centronics benannt ist. Die Original-Version hatte allerdings eine heutzutage am Rechner nur noch selten zu findende 36polige Centronics-Buchse (an den Druckern kann man diese Steckerausführung in der Regel noch bewundern). Da diese Buchse relativ teuer ist, wichen Lowcost-Hersteller auf 26polige Pfostenstift-Reihen aus (wie LPT2 bei der UniRAM-Karte) oder – noch billiger – gar auf 34polige Platinen-Direktstecker (Schneider/Amstrads erste CPC-Serie), die man den Floppy-Laufwerken entliehen hatte. Big Blue reduzierte beim PC den Anschluß auf 25polig Submin-D und setzte so dank des ihr eigenen Kompatibilitäts-Drucks den „Drucker-Industrie-Standard", an den sich später Atari und Commodore ankoppelten. Letztere litten allerdings beim Amiga 1000 unter Geschlechtsverwirrung und verwechselten männlich (Stifte) mit weiblich (Buchse). Den anderen Amigas ordneten sie jedoch das richtige, kompatible Geschlecht zu.

Bei der Reduktion von 36 auf 25 Pole gingen jedoch keine Signale verloren, man verzichtete lediglich auf einige abschirmende Masseleitungen. Zwischen zwei Daten- und Signalleitungen sollte nämlich eine Masseader liegen, damit man etwa zehn Fuß (also rund drei Meter) zwischen Rechner und Drucker problemlos überbrücken kann. Bei der Submin-D-Ausführung sind einige „statische" Signale (/Auto Feed, /Fault, /Reset, /Selekt In) zwischen die Datenleitungen gelegt, welche ihren Pegel in der Regel während eines Transfers nicht ändern und somit gleichfalls abschirmend wirken.

Der Signalpegel ist bei allen Centronics-Schnittstellen grundsätzlich auf TTL-Niveau. Vorsicht ist nur bei älteren Druckern geboten, die mit sehr kleinen Pull-up-Widerstandswerten die heutigen LS- oder CMOS-Treiber überlasten.

Bei PS/2, Atari und Amiga ist die Druckerschnittstelle bidirektional betreibbar, ebenso bei einigen PC- und AT-Druckerports (zum Beispiel auf der UniRAM-Karte). Man hat dann einen generellen interrupttauglichen I/O-Port für Messen, Steuern, Regeln oder schnellen Datentransfer. Die Transfer-Rate ist allerdings in der Regel nach oben hin durch Abblockkondensatoren (2,2 nF bei PC und PS/2) auf den Datenleitungen beschränkt.

Serielle Schnittstelle

Die asynchrone serielle Schnittstelle ist zuerst 1969 von der EIA (Electronic Industries Association) unter der Bezeichnung RS-232 definiert worden. Nach einigen „Vorversionen" ist letztendlich die C-Version (RS-232-C) allgemein bekannt geworden, die neben dem funktionellen Ablauf auch die elektrischen Kennwerte (Signalpegel, Kurzschlußfestigkeit, Abschlußwiderstand) festlegte. Diese amerikanische Norm nahm die für solche Fragen zuständige Internationale Kommission CCITT (Comité Consultatif International Télégraphique et Téléfonique) auf und standardisierte unter der Bezeichnung V.24 den funtkionellen Ablauf einer asynchronen bitseriellen Übertragung sowie die Bepinnung. Die konkreten elektrischen Anforderungen wurden in eine zweite Norm namens V.28 verbannt, so daß erst V.24 und V.28 zusammen der RS-232-C entsprechen. Das deutsche Pendant dazu ist die DIN 66020.

Um vollständig RS-232-C- oder V.24/28-kompatibel zu sein, ist eine Vielzahl von Signalleitungen nötig. Darunter sind solche, die die gewählte Übertragungsgeschwindigkeit und Sendeschrittrate dem jeweiligen Gegenüber mitteilen. Über einen zweiten Hilfskanal kann man mit verschiedenen Bitraten in Hin- und Rückrichtung arbeiten, was zum Beispiel für Btx gebraucht wird. So kamen die Normierer auf insgesamt 20 Signale was mit zwei Test- und drei Reserveleitungen den 25poligen Anschluß ergab, gemeinhin in Submin-D-Ausführung (mal weiblich, mal männlich).

IBM hielt sich beim PC zwar an die Stecker-Spezifikation (25 Submin-D, männlich), verzichtete jedoch auf die Spezial-Signale und den Hilfskanal. Statt dessen belegten die IBM-Entwickler die Reserve- und Testleitungen eigenmächtig mit den Anschlüssen für eine Stromschleife. Diese früher öfter eingesetzte Zweidraht-Verbindung (Fernschreiber) arbeitet nur mit TTL-Pegel und kann mit Klingeldraht auch größere Entfernungen sicher bewerkstelligen (RS-232-C ist „nur" für 100 Fuß, also 30 m, spezifiziert).

Ähnlich eigenmächtig verfuhr das Amiga-Team, das bei der 500/2000-Seite dort Tonsignale und Versorgungsspannungen aus dem Rechner führte. Das schwarze Schaf der Familie, der Amiga 1000, versorgt seinen weiblichen Submin-D-Port mit einer Fülle von abweichenden Signalen, so daß wer weiß was passieren kann, wenn man ein voll RS-232-C-kompatibles Datenendgerät daran anschließt.

Atari ging diesbezüglich kein Risiko ein und ließ die meisten Pins der 25poligen Buchse unbelegt. Solche Pin-Verschwendung war jedoch den IBM-Entwicklern ein Dorn im Auge, und da sie beim AT eh auf die antiquierte Stromschleife verzichteten, pfiffen sie auf die überholte Kompatibilität zum PC-Vorgänger und zu RS-232-C und kreierten den praktischen 9poligen Anschluß ... um beim PS/2 wieder zu den normgerechten – weitgehend unbelegten – 25 Polen zurückzukehren.

Der Mac glänzt durch eine oder zwei RS-422-Schnittstellen, die durch ihre Differential-Datenleitungen eine hohe Übertragungsrate von 231 kBd zulassen. Ältere Macs (bis zum 512e) vermitteln sich über eine 9polige Submin-D-Buchse (weiblich), während bei neueren Modellen die bei Selbstlötern so unbeliebten Mini-DIN-Buchsen zu finden sind. Legt man die invertierenden Empfangsdatenleitungen (RxD-) auf Masse und läßt TxD-unbeschaltet, so lassen sich die RS-422-Schnittstellen durchaus mit der RS-232-Norm verbinden. Besonders interessant: Der HSKi-Pin (Handshake In) läßt sich als externer Clock-Eingang umfunktionieren. Speist man dort 1 MHz ein, sendet und empfängt der Mac MIDI (entsprechende Software vorausgesetzt).

Trotz aller Normen macht gerade die asynchrone serielle Schnittstelle mit ihren diversen Handshakes immer wieder Schwierigkeiten, zumal die Spezifikationen sich nur auf die Verbindung Rechner zu Datenendgerät (Drucker, Modem) beziehen, Rechner-Rechner-Verbindungen (mit zu überkreuzenden Leitungen, sogenannten Null-Modems) waren nicht vorgesehen.

Tastatur

Zu CP/M-Zeiten wurden die Tastaturen manchmal über einen parallelen Port, meist aber über eine asynchrone serielle Schnittstelle betrieben (ASCII-Tastatur). Letzteres findet man heutzutage zum Beispiel beim Atari. Andere Tastaturen takten die seriellen Daten über eine zusätzliche CLK-Leitung aus. Dazu sind inklusive Spannungsversorgung vier Leitungen hinreichend, mehr haben Apple, Commodore und IBM auch nicht vorgesehen – wenngleich IBM noch eine Reserve-Leitung eingeplant hat. Je nachdem, wer die CLK-Leitung bedient, kann man Daten auch vom Rechner zur Tastatur (genauer: zum dortigen Tastatur-Prozessor) senden.

PC/AT-Tastaturen (und viele ASCII-Tastaturen) finden im allgemeinen über eine 5polige DIN-Buchse Eingang zum Rechner, die modernen Varianten davon sind die amerikanischen Telefon-Stecker oder beim PS/2 die 6poligen Miniatur-DIN-Buchsen. Eine Multifunktionstastatur (MF2) kann man über einen

Druckerschnittstelle

Rechner		PC, AT, PS/2, Atari	Amiga 1000	Amiga 500 2000/3000	Archimedes
25pol. Subm-D	36pol. Centr.	25-Subm-D weiblich	25-Subm-D männlich	25-Subm-D weiblich	25-Subm-D weiblich
Bild	(10)	(6)	(5)	(6)	(6)
Pins					
1	1	/Strobe	/Strobe	/Strobe	/Strobe
2	2	D1	D1	D1	D1
3	3	D2	D2	D2	D2
4	4	D3	D3	D3	D3
5	5	D4	D4	D4	D4
6	6	D5	D5	D5	D5
7	7	D6	D6	D6	D6
8	8	D7	D7	D7	D7
9	9	D8	D8	D8	D8
10	10	/ACK	/ACK	/ACK	/ACK
11	11	BUSY	BUSY	BUSY	BUSY
12	12	PE	PE	PE	–
13	13	Select	Select	Select	–
14	14	GND	GND	+5 V	–
15	32	GND	GND	–	–
16	31	GND	GND	/RESET	–
17	36	GND	GND	GND	GND
18	19..31	GND	GND	GND	GND
19		GND	GND	GND	GND
20		GND	GND	GND	GND
21		GND	GND	GND	GND
22		GND	GND	GND	GND
23		GND	+5 V	GND	GND
24		GND	–	GND	GND
25		GND	/RESET	GND	GND ❶

Mac SCC

Serial Communication Connector, RS-422

	Mac 128K, 512K(e) 9-Subm-D männlich	Mac Plus, SE, II 8pol. Mini-DIN
Bild	(3)	(11)
Pins		
1	GND	HSKo
2	+5 V	HSKi
3	GND	TxD–
4	TxD+	GND
5	TxD–	RxD–
6	+12 V	TxD+
7	HSKi	–
8	RxD+	RxD+
9	RxD–	

❷

RS-232-C Serielle Schnittstelle

Rechner		PS/2 Atari	PC	Amiga 1000	Amiga 500 2000/3000	AT Archimedes (RS-423)
Buchse		25-Subm-D männlich	25-Subm-D männlich	25-Subm-D weiblich	25-Subm-D männlich	9-Subm-D männlich
Bild		(5)	(5)	(6)	(5)	(3)
Pins						
1	Schutzerde	–	–	–	–	DCD
2	Sendedaten	TxD	TxD	TxD	TxD	RxD
3	Empfangsdaten	RxD	RxD	RxD	RxD	TxD
4	Sendeteil ein	RTS	RTS	RTS	RTS	DTR
5	Sendebereitschaft	CTS	CTS	CTS	CTS	GND
6	Betriebsbereitschaft	DSR	DSR	DSR	DSR	DSR
7	Signal-/Betriebserde	GND	GND	GND	GND	RTS
8	Empfangssignalpegel	DCD	DCD	DCD	DCD	CTS
9	Modemtest	–	Stroms O	–	+12 V	RI
10	Modemtest	–	–	–	–12 V	
11	–	–	Stroms O	–	Ton O	
12	Rückkanal Empf.-pegel	–	–	–	–	
13	Rückkanal Sendebereit	–	–	–	–	
14	Rückkanal Sendedaten	–	–	–5 V	–	
15	Sendeschrittakt von DCE	–	–	Ton O	–	
16	Rückkanal Empf.-dat.	–	–	Ton I	–	
17	Empfangsschrittakt	–	–	EB	–	
18	–	–	Stroms I	/INT2	Ton I	
19	Rückkanal Sendeteil ein	–	–	–	–	
20	Terminal betriebsbereit	DTR	DTR	DTR	DTR	
21	Empfangsgüte	–	–	+5 V	–	
22	Ankommender Ruf	RI	RI	–	RI	
23	Übertragungsgeschw.	–	–	+12 V	–	
24	Sendetakt von DTE	–	–	3,58 MHz	–	
5	–	–	Stroms I	/RESB	–	

❸

Tastatur

	ASCII	PC/AT	Atari Mega-ST	Mac 128K 512K(e)	Mac SE,II	Amiga 1000	Amiga 500	Amiga 2000/3000	Archie	PS/2
	5pol. DIN	5pol. DIN	6pol. RJ-11	4pol. RJ-11	4pol. Mini-DIN	4pol. RJ-11	8pol. intern	5pol. DIN	6pol. Mini-DIN	6pol. Mini-DIN
Bild	(2)	(2)	((8))	(8)	(12)	(8)	–	(2)	(14)	(14)
Pins										
1	/DATA	KBCLK	+5 V	GND	ADB	+5 V	CLK	CLK	Reset	KBDATA
2	DATA	KBDATA	+5 V	CLK	–	KBCLK	KBDATA	KBDATA	–	–
3	/RESET	–	TxD	DATA	+5 V	KBDATA	+5 V	–	GND	GND
4	GND	GND	RxD	+5 V	GND	GND	–	GND	+5 V	+5 V
5	+5 V	+5 V	GND				GND	+5 V	RxD	KBCLK
6	Abschirm	Abschirm	GND				/RES	Abschirm	TxD	–
7							Pow-LED		Abschirm	
8							Drv-LED			

❹

einfachen Adapter („nur das Kabel löten") wahlweise an DIN oder Mini-DIN anschließen, die Tastatur paßt sich automatisch an den Rechner an [1].

PC- und AT-Tastaturen fahren allerdings ein unterschiedliches Protokoll, so daß hier kein Austausch möglich ist. Mit etwas Hardware lassen sich PC- oder MF2-Tastaturen auch am Atari anschließen [2].

Manche PCs (Schneider/Amstrad 1512) haben einen kombinierten Tastatur-/Maus-Anschluß. Des weiteren findet man auch Tastaturen mit integriertem Trackball und vieles mehr.

Maus und Joystick

Jeder Rechner, der etwas auf sich hält, bietet inzwischen einen dedizierten Maus-Anschluß. Vorbei sind die Zeiten, wo die Maus eine serielle Schnittstelle blockierte, wie es bei der Microsoft-Maus noch üblich war. Bei den älteren Macs endete der Mäuseschwanz in einer 9poligen Submin-D-Buchse. Ebenso – natürlich mit völlig anderer Belegung und Funktion – beim Atari, wobei man hier die Maus wahlweise durch einen Joystick ersetzen kann. Für einen zweiten Joystick halten Atari und Amiga eine weitere 9polige Submin-D-Buchse bereit. Bei PC/AT bietet eine Game-Karte den spielfreudigen Benutzern den Anschluß zweier Joysticks ebenfalls über Submin-D, diesmal 15polig, an.

Sowohl Apple wie auch IBM haben bei ihren neueren professionellen Kreationen natürlich auf Joysticks verzichtet – dafür mehr auf gleichgestaltete Tastatur- und Mausanschlüsse geachtet. Die PS/2-Modelle weisen hierfür jeweils 5polige Mini-DIN-Buchsen auf; Apple war mit den Pins noch sparsamer und beschränkt sich bei den neueren Rechnern vom SE aufwärts auf 4polige Mini-DIN-Buchsen, über die ein bidirektionaler serieller Datenverkehr (Apple Desktop Bus, ADB) mit recht aufwendigem Protokoll abgewickelt wird. Maus und Digitalisiertablett liefern ihre Positionsdaten ebenfalls über den ADB.

Floppy

Floppy-Anschlüsse werden gemeinhin mit dem Namen Shugart in Verbindung gebracht – 34poliger Platinenstecker bei 5 1/4" oder Pfostenstecker bei 3 1/2" –, doch auch hier sind diverse Spielarten möglich [3].

Der gute alte PC hatte noch einen externen 37poligen Anschluß für externe Laufwerke aufzuweisen, mit Shugart-ähnlicher Belegung. Schneiders CPCs kamen gar mit einem gespiegelten Shugart-Bus daher. Atari liefert entsprechende Signale über einen 14poligen Rundstecker, Amiga über 23polige Submin-D. Beide Rechner beschränken die Anzahl der Select-Leitungen auf das Notwendigste. AT und PS/2 haben keine externen Floppies vorgesehen, intern sind Standard-Floppystecker vorhanden, die beim PS/2 um die Stromversorgung erweitert sind. Will man beim PS/2 „normale" 5 1/4"-Laufwerke anschließen, so hat man noch an einigen Signalleitungen Pull-up-Widerstände anzubringen [4].

Die Macs haben mit Shugart nichts am Hut, sondern lehnen sich vom Layout und den Signalnamen her an die vom Apple II bekannte Belegung an, auch wenn an die 19polige Submin-D-Buchse noch nicht einmal Apple-II-5 1/4"-Drives angeschlossen werden dürfen.

❺

Maus

	Atari 9-Subm-D männlich	Amiga 9-Subm-D männlich	Mac SE,II 4pol. Mini-DIN	Mac 128,512(e) 9-Subm-D weiblich	Archimedes 9pol. Mini-DIN
Bild	(3)	(3)	(12)	(4)	(13)
Pins					
1	hoch/XB	hoch/V	ADB	GND	X-Referenz
2	runter/XA	runter/H	–	+5 V	Taste 1
3	links/YA	rechts/VQ	+5 V	GND	Taste 2
4	rechts/YB	links/HQ	GND	X1 Quadratur	GND
5	–	–		X2 Interrupt	X-Richtung
6	Taste li.	Taste li.		–	+5 V
7	Abschirm	+5 V		Taste	Y-Referenz
8	GND	GND		Y1 Quadratur	Taste 3
9	Taste re.	Taste re.		Y2 Interrupt	Y-Richtung

Floppy

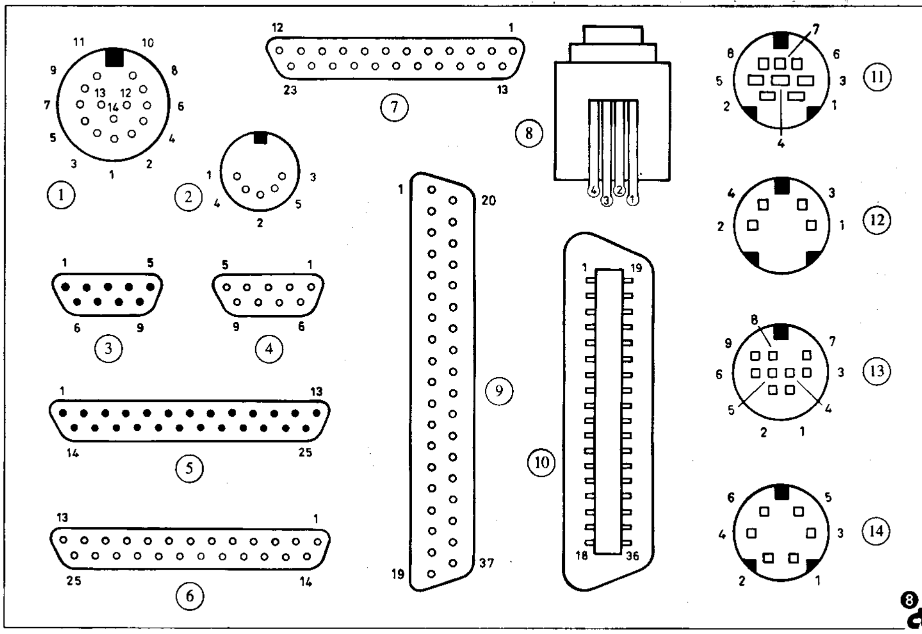
	Shugart Pfosten, Platine	PC 37-Subm-D weiblich	Atari 14pol. DIN	Amiga 23-Subm-D, weiblich 500/3000	Amiga 23-Subm-D, weiblich 2000B
Bild		(9)	(1)	(7)	(7)
Funktion					
Head-Load	2	–	–		
(HD/LD-Density)	2)	–	–		
In Use	4	–	–		
(Head-Load	4)	–	–		
DS3	6	8/C	–	20	9
(Ready	6)	–	–	–	–
Index	8	6	4	22	22
DS0	10	–	5	–	–
DS1	12	–	6	21	–
DS2	14	9/D	–	9	21
Motor on	16	7/C,10/D	8	8	8
Direction in	18	11	9	19	19
Step	20	12	10	18	18
Write Data	22	13	11	17	17
Write Gate	24	14	12	16	16
Track 0	26	15	13	15	15
Write Protect	28	16	14	14	14
Read Data	30	17	1	2	2
Side Select	32	18	2	13	13
Ready	34	–	–	1	1
(Disk Change	34)	–	–	11	11
(In Use	34)	–	–	–	–
D-Reset /DRES	–	–	–	10	10
+5 V	–	–	–	12	12
+12 V	–	–	–	23	23
GND	1,3..33	19..37	3,7	3..7	3..7

❻

SCSI-Anschluß

SCSI 50polig	Funktion	Mac 25-Subm-D weiblich	Amiga 3000 25-Subm-D männlich
Bild		(6)	(5)
Pins			
2	D0	8	8
4	D1	21	21
6	D2	22	22
8	D3	10	10
10	D4	23	23
12	D5	11	11
14	D6	12	12
16	D7	13	13
18	DBP Parity	20	20
20	GND	7	7
22	GND	9	9
24	GND	14	14
26	Term Power	25	25
28	GND	16	16
30	GND	18	18
32	ATN	17	17
34	GND	24	24
36	BSY	6	6
38	ACK	5	5
40	RST	4	4
42	MSG	2	2
44	SEL	19	19
46	C/D	15	15
48	REQ	1	1
50	I/O	3	3
1,3..49	GND		

7



8

SCSI

SCSI (Small Computer System Interface) wird immer mehr zum Anschlußstandard für Festplatten, Streamer, Scanner und ähnliche Geräte. Neuerdings scheint auch IBM nach einen Ausflug zu ESDI wieder mehr auf SCSI zu setzen. Während Mac und der neue Amiga 3000 von Haus aus SCSI-Bus fahren, reichte es beim Atari nur für einen schlappen Abklatsch namens DMA-Port. Mit wenig (Software-)Aufwand läßt sich eine handelsübliche und preiswerte SCSI-Platte an den Mac [5] oder mit etwa Hardware-Hilfe an den Atari [6] anschließen. Industrie-Kompatible dagegen benötigen für SCSI-Kompatibilität spezielle Steckkarten, sogenannte Host-Adapter.

Der nächste Teil unserer Kartei behandelt Stromversorgungsanschlüsse, Videoverbindungen und „Spezialitäten" verschiedener Rechner.

Literatur

[1] Knöpfchen, Knöpfchen, PC-Tastaturen auf den Zahn gefühlt, c't 6/88, S. 148 und c't 7/88, S. 178
[2] PC-Tastatur am Atari ST, eine Low-Cost-Lösung für versierte Bastler, c't 10/88, S. 208 und c't 11/88, S. 150
[3] Anschluß gesucht, Floppy-Laufwerke zum Laufen gebracht – vom sogenannten Shugart-Bus und ähnlichen „Standards", c't 7/87, S. 169
[4] Große Scheiben für PS/2, 5,25"-Laufwerk am IBM PS/2 Modell 50/60/80, c't 9/88, S. 166
[5] Mac spielt harte Scheiben, Preiswerte Festplatte am Apple Macintosh +/SE/II, c't 2/89, S. 166 (siehe auch c't 3/89, S. 12)
[6] Die Harddisk-Alternative, SCSI und Links für den Atari ST, c't 1/90, S. 166
[7] Mac am Draht, Kommunikation mit dem Apple Macintosh, c't 3/89, S. 170

wird fortgesetzt

Nachdruck aus c't – Magazin für Computertechnik, Heft 8/1990

Wegbereiter der Informatik

JOHANN (JOHN) LUDWIG v. NEUMANN

∗ 1903 Budapest
† 1957 Washington

J. v. Neumann entstammt einer österreich-ungarischen Bankiersfamilie. Schon früh zeigte sich bei ihm eine besondere mathematische Begabung, und noch während seiner Gymnasialzeit erhielt er Privatunterricht in Mathematik von einem Assistenten der Budapester Universität. Bereits vor Vollendung seines 18. Lebensjahres erschien seine erste mathematische Arbeit über die Lage der Nullstellen gewisser Minimalpolynome. Nach dem Abitur begann er an der Budapester Universität ein Mathematikstudium, hielt sich aber vorwiegend an der ETH Zürich auf, wo er zusätzlich Chemie studierte. Auch an der Berliner Universität hörte er Mathematik- und Physikvorlesungen. Schließlich legte er in Zürich ein Examen als Diplomingenieur der Chemie ab und promovierte kurz danach (1926) an der Budapester Universität bei L. Fejér im Fach Mathematik.

In den Jahren 1927–29 wirkte er in Berlin an der Universität als Privatdozent. Nachfolgende Wirkungsstätten waren Hamburg und Los Alamos. Im Jahre 1931 wanderte er in die USA aus und wurde 1933 in Princeton Professor für Mathematik.

J. v. Neumann gilt als ein Begründer der modernen Funktionalanalysis, hat aber auch auf zahlreichen anderen Gebieten der modernen Mathematik grundlegende Untersuchungen durchgeführt (Algebra, Mengenlehre, Beweistheorie, Maßtheorie, allgemeine Topologie). Eine seiner wahrscheinlichkeitstheoretischen Arbeiten (1928) wurde zum Ausgangspunkt einer umfangreichen Spieltheorie; in ihr wird eine weitgefaßte Definition des Spielbegriffs gegeben, die auch Modelle des Wirtschaftslebens umfaßt. Des weiteren ist er an einem neuen axiomatischen Aufbau der Mengenlehre mit beteiligt, der sich von dem stufentheoretischen grundsätzlich unterscheidet und nicht – wie jener – zu Antinomien führen kann. Auch zur Anwendung der modernen Mathematik in der Physik (Quantentheorie) und der Psychologie hat er Beiträge geleistet.

Insbesondere aber ist J. v. Neumann maßgeblich an der Entwicklung der elektronischen Computertechnik beteiligt gewesen, er hat in den USA die ersten Großrechenanlagen mit konzipiert. In den Jahren 1945–55 war er Direktor eines Projektierungsbüros für elektronische Rechenmaschinen. Auf ihn geht das Konzept der internen Programmspeicherung zurück; unabhängig von Konrad Zuse hat er das grundlegende Bauprinzip des klassischen, seriell arbeitenden Digitalrechners mitgeschaffen. Die eingehende Beschäftigung mit Theorie und Praxis der elektronischen Rechenautomaten führten ihn zum einen zu Problemen der mathematischen Logik und numerischen Analysis, zum anderen zu einer Theorie der idealisierten Nervensysteme. Sein nachgelassenes, zum Teil noch auf dem Krankenlager geschriebenes Buch zu letzterem Fachgebiet trägt den Titel „The Computer and the Brain" (Rechenautomat und Gehirn).

J. v. Neumann hat auch maßgeblich an der technischen Entwicklung der Atomkernspaltung mitgewirkt.

Dr. Klaus Biener

Lisp

Teil 3

Rainer Rosenmüller

Dieser Teil des Lehrgangs behandelt die Iteration – für alle, die nach dem streng rekursiven Teil 2 bei der etwas konventionelleren Iteration Abwechslung suchen –, datengesteuerte Programmiertechniken, um die Vorteile von Lisp voll ausschöpfen zu können, die Ein- und Ausgabe sowie Readmakros und Makros zum Schreiben lesbarer Programme. Außerdem gibt es für die Leser(innen), die ihre Termine ohne Computer nicht mehr in den Griff bekommen, die Fortsetzung des Terminkalenderprogramms.

Datengesteuerte Programmierung

Die Iteration

Die *natürliche* Programmiermethode in Lisp ist die Rekursion. Wie in allen Programmiersprachen wird aber auch in Lisp die Iteration durch bestimmte Sprachelemente unterstützt. Die am häufigsten benutzten sind **Dotimes** und **Dolist**. Dotimes ist ein einfacher Zyklus über ganze Zahlen, wie es ihn in allen Sprachen gibt:

(dotimes (<var> <countform> [<resultform>]**)** . <progbody>**)**

Die Form Countform wird ausgewertet und muß eine ganze Zahl als Endwert liefern. Dann läuft die Variable Var von einschließlich Null bis ausschließlich des berechneten Endwerts. Mit jeder dieser Variablenbelegung werden dann die Formen in Progbody abgearbeitet. Zum Schluß wird das Endergebnis durch die Auswertung von Resultform bestimmt. Ist diese nicht angeben worden, so ist das Ergebnis NIL. Ist der Endwert Null oder negativ, so wird der Körper nicht abgearbeitet. Die zu Dotimes analoge Lispform ist Dolist:

(dolist (<var> <listform> [<resultform>]**)** . <progbody>**)**

Dabei wird Listform als erstes ausgewertet und muß eine Liste ergeben. Danach nimmt die Variable Var nacheinander die Werte aller Elemente der Liste an, und der Programmkörper wird mit dem jeweils aktuellen Wert abgearbeitet. Damit kann man die in Bild 22 angegebene – allerdings nicht sehr elegante – Variante der schon bekannten Funktion Reverse schreiben. Diese Konstrukte sind in ihrer Argumentliste sehr einfach und genügen nicht immer den Anforderungen, die sich aus komplexeren Aufgabenstellungen ergeben. Deshalb gibt es noch eine weitere, besser lesbare Form, die auch denen der konventionellen Sprachen sehr nahe kommt:

(do ({(<var> [<initial-value> [<step>]])}**)**
(<endtest> {<result-form>})
{<body-form>}**)**

Die Variablen werden durch die Initialisierungsformen bzw. durch NIL initialisiert und bei jedem Durchlauf durch die Form Step weitergestellt. Ist die Bedingung Endtest erfüllt, werden die Resultformen abgearbeitet und die Schleife wird mit dem Ergebnis der letzten Form verlassen; im anderen Fall werden die Body-Formen abgearbeitet (Bild 23).

```
(defun mp-reverse2 (list)
       (do ((templist list (cdr templist))
            (res nil (cons (car templist) res)))
           ((null templist) res)
           "der koerper ist leer"))
```

Bild 23

Weiterhin gibt es die Form Do∗, wobei die Bindung der Variablen nicht mit Let, sondern mit Let∗ realisiert wird.

Einen zweiten Typ der Iterationskonstrukte stellt Loop dar. Loop ist in seiner Form nicht so festgelegt, wie das oben beschriebene Do. Seine Variabilität erfordert aber auch etwas mehr eigene Programmierung.

(loop {<form>}**)**

Die Formen, die im Körper der Loop-Anweisung stehen, werden wiederholt sequentiell abgearbeitet. Verlassen wird die Loop-Anweisung durch ein Return. Achtung! In Mulisp hat Loop eine etwas exotische Syntax.

Die Funktion Reverse sieht dann wie in Bild 24 dargestellt aus. Bei all diesen Formen handelt es sich jedoch nicht um Funktionen, sondern um Spezialformen bzw. Makros.

```
(defun mp-reverse3 (list)
  (let ((res nil))
       (loop (when (null list) (return res))
             (setq res (cons (car list) res))
             (setq list (cdr list)))))
```

Bild 24

Bevor gezeigt wird, wie Sie sich selbst Makros schreiben können, möchte ich Ihnen noch die explizite Auswertung und einige lispspezifische Iterationsformen erläutern.

Die explizite Auswertung

Die strukturelle Gleichheit von Daten und Programmen macht eine einfache automatische Konstruktion von Programmen möglich und erleichtert das Schreiben von programmerzeugenden Programmen. Die so erzeugten Programme können beispielsweise Werte von Symbolen sein. Die Abarbeitung solcher Programme wird erst durch den expliziten Aufruf der Funktion **Eval** angewiesen. Eval wertet den Ausdruck in der aktuellen dynamischen und in einer leeren lexikalen Umgebung aus. Beinhaltet beispielsweise die Variablen a das *Programm* (+ 1 2 3), so können Sie dieses wie folgt benutzen.

(setq a '(+ 1 2 3)) ==> (+ 1 2 3)
(cons '(das ergebnis ist) a)
==> ((DAS ERGEBNIS IST) + 1 2 3)
(cons '(das ergebnis ist) (eval a))
==> ((DAS ERGEBNIS IST) . 6)

Hierbei müssen sie aber beachten, daß das Argument von Eval zweimal ausgewertet wird. Zuerst wertet der Interpreter (wie für jede Funktion) das Argument aus. Danach verlangt die Funktion Eval, das das übergebene Argument noch einmal ausgewertet wird.

Ein großer Vorteil der Sprache Lisp ist darin zu sehen, daß man mit nur geringem Aufwand erweiterbare Funktionen schreiben kann. Wollen Sie beispielsweise eine Funktion *Typep* schreiben, die prüfen soll, ob ein Lisp-Objekt zu einem bestimmten Datentyp gehört, so müßten Sie diese Funktion ständig ändern, weil Sie sich mit Deftype und Defstruct ständig neue Typen definieren können. Schreiben Sie sich aber die Funktion Typep so, daß sie durch Defstruct oder anderen Funktionen physisch erweitert wird, so könnten Sie Typep nie kompilieren. Diesen Nachteil, den man mit der Erweiterbarkeit einer Funktion in Kauf nehmen muß, kann man durch die datengesteuerte Programmierung umgehen. Wenn Sie in der P-Liste des Symbols Typep unter dem Indikator des Typs die Typprüffunktion ablegt, so könnten die oben genannten Funktionen diese Liste ständig erweitern und das Problem wäre gelöst. Die dafür noch benötigte Funktion heißt Funcall und hat folgende allgemeine Form:

(funcall <fun-object> <args>**)**

Diese Funktion ruft die Funktion Fun-objekt auf und übergibt ihr die restlichen Argumente als aktuelle Parameter. Die P-Liste von Typep könnte etwa den folgenden Inhalt haben:

(string #'stringp stream #'streamp
ship #'(lambda (objekt)
(and (structurep object)
(eq (car object) 'ship))) ...)

Und so könnte Typep folgendermaßen definiert werden:

(defun mp-typep (object type)
(funcall (get 'typep type) object))

Damit ist die Funktion Typep natürlich noch nicht vollständig definiert, aber ein wesentlicher Teil ist schon geschafft. Das in der P-Liste auftretende #'streamp ist eine Abkürzung für (function streamp). Nur liefert *Function* (#') im Unterschied zu Quote (') die funktionale Bedeutung des Symobols, einschließlich der für die Funktion notwendigen Lisp-Umgebung, und nicht das Symbol selbst als Ergebnis. Werden die Argumente einer Funktion nicht einzeln, sondern bereits in einer Liste übergeben, müssen Sie allerdings anstelle von Funcall die Funktion Apply benutzen. Die allgemeine Form lautet:

(apply <fun-object> <arg> <more-args>**)**

Diese Funktion arbeitet ähnlich wie Funcall; das letzte Argument kann aber auch eine Liste sein, die an die Liste der davor angegebenen Argumente angehängt wird.

(apply #'+ 3 4 '(6 8)) ==> 21

Die Lambda-Funktion und die Lambda-Liste

In der P-Liste von Typep ist aber noch eine weitere, unbenannte Funktion enthalten. Die allgemeine Form dafür ist:

```
(defun palindromp (string)
   (let ((start 0) (end (length string)))
        (dotimes (i (floor (- end start) 2) t)
                 (unless (char-equal (char string (+ start i))
                                     (char string (- end i 1)))
                        (return nil)))))

(palindromp "Marktkram") ==> T
(palindromp "Das ist kein Palindrom") ==> NIL
```

Bild 21

```
(defun mp-reverse1 (list)
   (let ((newlist nil))
        (dolist (element list newlist)
                (setq newlist (cons element newlist)))))
```

Bild 22

(lambda <lambda-list> {<declaration>} {<form>}**)**
Die allgemeine Form der Lambda-Liste zeigt Bild 25.

```
({<required-var>}
 [&optional {<optional-var> |
              (<optional-var> <initial-value>)}]
 [&rest <rest-var>]
 [&key
      {<key-var> |
      ({<key-var> |
      (<keyword> <key-var>)} <initial-value>)}])
```

Bild 25

Wenn solch eine Form auf Argumente angewendet wird, so werden erst die Variablen der Lambda-Liste lexikalisch gebunden, und anschließend werden die Formen sequentiell abgearbeitet. Das Ergebnis ist das der letzten Form. Die Bindung der Variablen des Variablenteils geschieht in folgender Reihenfolge: Zuerst werden die Variablen der obligaten Parameter gebunden. Danach werden die Variablen der optionalen Parameter an die angegebenen Werte (die Werte der Initialisierung bzw. NIL, wenn keine Anfangswerte gegeben wurden) gebunden. Anschließend wird die Restvariable an die Liste der noch verbleibenden Argumente gebunden. Die Schlüsselwortvariablen werden zum Schluß gebunden. Diese sind keine Stellungsparameter, und die Zuordnung Variable-Wert erfolgt durch die paarweise Angabe Schlüssel-wort-Wert. Unbenannte Funktionen können an allen Stellen stehen, an denen Funktionen verlangt werden.
((lambda (a b) (∗ a b)) 4 5) ==> 20

Die Map-Funktionen

Diese Funktionen sind die lisptypischen Iterationskonstrukte. Zu der Gruppe gehören Mapc, Mapcan, Mapcar, Mapl, Maplist und Mapcon; ihre allgemeine Form sieht folgendermaßen aus:
(mapxxx <function> <list> <mor-lists>**)**
Allen Funktionen ist gemeinsam, daß sie die übergebene Funktion (1. Argument) auf ihre Argumentliste(n) anwenden und daß sie die Argumentliste(n) nach jeder Anwendung auf den Cdr der Argumentliste(n) reduzieren. Es müssen so viele Argumentlisten angegeben werden, wie die Funktion (1. Argument) als Argumente benötigt. Die Wirkungsweise der Map-Funktionen läßt sich nach der Art der Argumentübergabe an die Funktion und der Ergebnisbildung unterscheiden.
Die Funktionen Mapc, Mapcar und Mapcan übergeben an ihre Argument-Funktion jeweils den Car der aktuellen Argumentlisten, während die Funktionen Mapl, Maplist und Mapcon jeweils die gesamte aktuelle Argumentliste übergeben.
Die Funktionen Mapc und Mapl ignorieren dabei die Ergebnisse der Iterationsschritte. Die Funktionen Mapcar und Maplist bilden eine Liste der Ergebnisse aller Iterationsschritte mittels Cons und die Funktionen Mapcan und Mapcon bilden eine Liste aller positiven Ergebnisse (ungleich NIL) der Iterationsschritte mittels Nconc. Die Ergebnisse von Mapc und Mapl hängen vom Lisp-Dialekt ab. In den meisten alten Dialekten ist der Wert NIL aber in Common Lisp ist er gleich dem zweiten Argument. Daraus ergeben sich die typischen Anwendungen der Funktionen: Mapc und Mapl werden zur Erzielung von Seiteneffekten benutzt, Mapcan und Mapcon werden als Filterfunktionen benutzt.
(Anmerkung: Die Funktion Mapl kann auch manchmal Map heißen!) Zum Beispiel kann man eine Liste der Elemente einer Ausgangsliste bilden, die einen Test erfüllen, indem man folgendes schreibt:
(mapcan #'(lambda (x)
(if (test x) (list x) nil)) liste1)
Man kann auch leicht zwei Listen zu Paaren mischen:
(mapcar #'cons '(1 2 3)
'(auswahl-1 auswahl-2 auswahl-3))
==> ((1 . AUSWAHL-1) (2 . AUSWAHL-2) (3 . AUSWAHL-3))

Einfache Eingaben

Die wichtigsten Eingabefunktionen sind Read-char, Read-line und Read. Für alle Funktionen in Common Lisp gilt die folgende Parameterliste:
&optional (<stream> nil**)**
(<eof-error-p> t) <eof-value>
(<recursive-p> nil)
Die hier angeführten Streams sind Lisp-Objekte, die die Verbindung zwischen Lisp und dem Betriebssystem des Rechners herstellen. Die Standardstreams sind beim Start des Systems bereits definiert, können aber vom Lispnutzer neu gebunden werden. Das Argument Stream kann somit NIL für den Wert der Variablen ∗STANDARD-INPUT∗ als Eingabe-Stream, T für ∗TERMINAL-IO∗ oder ein beliebiger Eingabe-Stream sein. Das Argument Eof-error-p zeigt an, ob eine Fehlerbehandlung ausgelöst werden soll, wenn während des Lesens das Dateiende (EOF) erreicht wurde. Ist Eof-erro-p gleich NIL, so wird bei Erreichen des Dateiendes Eof-value als Ergebnis des Lesevorgangs zurückgegeben. Das Ergebnis der Funktion Read-char ist ein einzelnes Zeichen (als Lisp-Objekt), das von Read-line ist eine Zeichenkette, die die Zeichen der nächsten Zeile des Streams (ohne New-line) enthält, und zusätzlich als zweiten Wert T oder NIL als Zeichen dafür, ob das abschließende Zeichen der Zeile ein EOF war oder nicht. Die Funktion Read dagegen liest den nächsten Lisp-Ausdruck (in der Lisp-Syntax unabhängig von der physischen oder logischen Strukturierung des Streams).
(read-char)x
==> #\x
(read-line)Das ist eine Zeile
==> "Das ist eine Zeile"
(read) (a b c #\Space 1 2.4 "otto")
==> (A B C #\Space 1 2.4 "otto")
Die Funktionen Read-char und Read-line können auch dazu benutzt werden, lispunabhängige, nutzereigene Eingaben einzulesen.

Einfache Ausgaben

Die wichtigsten Ausgabefunktionen lauten in ihrer allgemeinen Form:
(prin1 <object> **&optional** <stream>**)**
(princ <object> **&optional** <stream>**)**
(terpri &optional <stream>**)**
Das Argument Stream hat die gleiche Funktion wie bei der Eingabe. Die Funktion Terpi (terminate print-line) gibt ein Newline aus. Die Funktion Prin1 gibt die externe Darstellung eines Lisp-Objektes in der vollständigen Lisp-Syntax aus, das heißt, diese Form kann durch Read wieder eingelesen werden, dagegen gibt Princ eine verkürzte, lesbare Darstellung aus.
(prin1 '(a #\b "abc\\xyz\"123")
==> (A #\b "abc\\xyz\"123")
(princ '(a #\b "abc\\xyz\"123")
==> (A b abc\xyz"123)
Als Ergänzung gibt es noch die Funktion Print. Sie gibt ein New-line aus, ruft dann Prin1 auf und gibt danach ein Leerzeichen aus. Manche Lisp-Dialekte haben noch die Funktionen Pprint für eine strukturierte Ausgabe der Lisp-Objekte und Format für die formatierte Ausgabe.
Alle Printfunktionen werden in ihrer Wirkung durch globale Steuervariablen weiter spezifiziert. So bestimmt zum Beispiel die Variable ∗PRINT-BASE∗, zu welcher Basis die rationalen Zahlen ausgegeben werden. Die Variablen ∗PRINT-LEVEL∗ und ∗PRINT-LENGTH∗ begrenzen die Tiefe und die Länge von auszugebenden Listen, wenn sie nicht den Wert NIL haben.
Ist ∗PRINT-LEVEL∗ eine positive ganze Zahl, so werden Listen nur bis zu dieser Tiefe ausgegeben. Alle tiefer gehenden Komponenten der Liste werden durch das Zeichen # angedeutet. Wird die Listenausgabe in der Länge begrenzt, so werden nur so viele Elemente ausgegeben, wie verlangt werden. Ist die Liste länger, so wird der Rest durch die Zeichen ... abgekürzt. Das kann sehr nützlich sein, wenn man große oder zyklische Listen ausgeben will, um sie zu überwachen oder zu identifizieren.
Das folgende Beispiel soll das verdeutlichen.
(setq ∗print-level∗ 2) ==> 2
(setq ∗print-length∗ 3) ==> 3
'(a (b c (d e) f) (g h i k) l m)
==> (A (B C # ...) (G H I ...) ...)

Die formatierte Ausgabe

Die allgemeine Form der formatierten Ausgabe lautet:
(format <stream> <control-string-> <args>**)**
Das Argument Stream hat hier eine etwas andere Bedeutung: T steht für ∗STANDARD-OUTPUT∗ und NIL bewirkt, daß von Format eine Zeichenkette erzeugt wird, die die ausgegebenen Zeichen enthält. Die Ausgabe entsteht folgendermaßen. Die Zeichen, die in control-string stehen, werden nacheinander ausgegeben. Wird das Zeichen Tilde ~ erkannt, so beginnt eine Formatanweisung. Diese Anweisung wird analysiert und ausgeführt, wobei Elemente aus der restlichen Argumentliste Args verbraucht werden können. Anschließend wird die Steuerzeichenkette weiter abgearbeitet. Die wichtigsten Formatanweisungen sind:

~%	Terpi wird ausgeführt
~s	Prin1 wird mit dem nächsten Argument aufgerufen
~a	Princ wird benutzt

(let ((x '(a b "xyz")))
(format t
"~%Das ist die Liste ~a mit~
~a Elementen~%third ist aber ~s!"
x (length x) (third x)))
==>
Das ist die Liste (A B xyz) mit 3 Elementen
third ist aber "xyz"!

Natürlich ist das nur eine sehr kurze Einführung in die Format-Welt, denn der Umfang

von Format übersteigt die Formatierungsunterstützungen aller anderen bekannten Sprachen.

Das Öffnen und Schließen von Dateien

Mit den bisher vorgestellten Funktionen ist nur eine Ausgabe auf den Bildschirm möglich, weil die Funktion Open, zur Erzeugung eines Streams noch fehlt. Diese hat für die Eingabe die allgemeine Form:
(open <filename> **:direction :input)**
Damit wird einen Stream erzeugt, der Eingaben (:input), Ausgaben (:output) oder Ein- und Ausgaben (:io) ermöglicht und der mit der angegebenen Datei verbunden ist. Der Name muß eine Zeichenkette mit für Dateinamen gültigen Zeichen sein, kann aber in Abhängigkeit vom Betriebssystem und des Lisp-Dialekts stark reduziert sein. Abgeschlossen wird ein Stream durch
(close <stream>**)**
Die Funktion Load zum Einlesen von Quelltexten könnte dann wie im Bild 26 aussehen.

```
(defun mp-load (filename &key (print nil))
  (format ";LOAD ~a start" filename)
  (do* ((str (open filename :direction :input))
        (obj (read str nil str)
             (read str nil str)))
       ((eq str obj)
        (format ";LOAD ~a end" filename)
        (return t))
       (if print
           (print (eval obj))
           (eval obj))))
```

***Bild** 26*

Am Anfang der Do∗-Schleife werden Str und Obj initialisiert, dann wird nur noch Obj in jedem Zyklus neu bestimmt. Interessant ist die Abbruchbedingung. Das Ende der Datei (EOF) wird vom Read erkannt und bewirkt, daß Read als Eof-value den Stream Str, der garantiert nicht eingelesen werden konnte, als Ergebnis liefert. Streams sind Datenobjekte, die keine externe Repräsentation besitzen und somit auch nicht in eine Datei ausgelagert werden können. Damit sind sie aber als eindeutige Markierung für das Ende einer Datei verwendbar und die Iteration kann bei diesem Ergebnis abgebrochen werden (Bild 27).

Aktion		Datei		Ergebnis
READ	==>	object1	==>	OBJECT1
READ	==>	object2	==>	OBJECT2
READ	==>		==>	#<STREAM 3FC4>

Bild 27

Makros und Readmakros

Makros ersetzen normalerweise ein Stück Quelltext durch einen anderen. In dem Read-eval-print-Zyklus gibt es in Lisp zwei Stellen, an denen solche Makros aktiv werden können, beim Read und beim Eval.
Readmakros sind einzelne Zeichen, die mit einer Lesefunktion verknüpft sind. Treten diese im Zeichenstrom auf, so setzt die zugehörige Funktion das Lesen fort und bestimmt auch das Ergebnis der Leseaktion. Die wichtigsten Readmakros sind das Quote-(') und das Backquote-Makro ('). Aus 'form wird beim Einlesen (QUOTE FORM) und aus 'form wird (BACKQUOTE FORM). Die Wirkung von Quote ist Ihnen ja bereits bekannt. Die Arbeitsweise des Backquote-Makros ist etwas komplizierter. Normalerweise wertet der Interpreter alle Formen aus; nur mit Quote kann eine Auswertung explizit unterdrückt werden. Innerhalb des Backquotes arbeitet das Lisp gewissermaßen im inversen Interpretermodus: Alle nicht extra markierten Ausdrücke werden als gequotet angesehen, soll eine Auswertung bestimmter Ausdrücke erfolgen, so muß dies explizit angegeben werden. Wie das Ergebnis der Auswertung in den Gesamtausdruck eingefügt werden soll, wird durch die Art der Markierung des Ausdrucks angegeben: Steht ein Komma davor, so wird das Ergebnis so wie es kommt eingebaut, steht ein ,@ davor, so werden die Elemente der bei der Auswertung entstehenden Liste in den Gesamtausdruck einzeln eingebaut. Damit eignen sich Backquote-Ausdrücke sehr gut zum Schreiben von Makros oder zur Erzeugung von Programmen.
(setq x '(∗ 2 3)) ==> **(∗ 2 3)**
'(list x y ,x z ,@x w)
==> **(list x y (∗ 2 3) z ∗ 2 3 w)**
Weitere Readmakros gibt es für Listen, für Zeichenketten, für Kommentare bis zum Zeilenende (;), für die schon bekannte Funktionsabkürzung (#'), für einfache Vektoren (#), für balancierte, schachtelbare Kommentare beliebiger Länge (#| ... |#), für Zeichen (#\\), für komplexe Zahlen (#C) und für binäre, oktale und hexadezimale rationale Zahlen (#B, #O und #X), die aber alle sehr dialektabhängig sind. Die meisten Readmakrozeichen sind terminierende Zeichen, das heißt, sie beenden ein angefangenes Lexem. So wird abc'def zu ABC und (QUOTE DEF) eingelesen. Unter den Standardmakrozeichen ist nur das Doppelkreuz (#) nicht terminierend und kann in ein Lexem eingebettet werden.
Die Auswertung der Makros erfolgt (im Gegensatz zu den Readmakros) beim Eval. Wenn der Interpreter eine Form auswertet, deren erstes Element in der Liste eine funktionale Bedeutung besitzt, so werden alle weiteren Objekte ausgewertet und anschließend der Funktion als Argumente übergeben. Liefert die Funktionsauswertung des ersten Elementes aber keine Funktion, sondern eine Spezialform oder ein Makro, so werden die weiteren Objekte unausgewertet an das Makro übergeben, das diese dann nach eigenem Ermessen auswertet oder nicht. In dieser ersten Auswertungsrunde liefert der Makro-Aufruf die sogenannte Makroexpansion. In der zweiten Auswertungsrunde wird nun der vom Makro erzeugte Kode ausgewertet und liefert das endgültige Ergebnis. Im Gegensatz zu anderen Sprachen (C, Assembler) werden in Lisp die Makros also zur Laufzeit ausgewertet. Die allgemeine Form zum Definieren eines Makros ist:
(defmacro <name> <lambda-list>
{<declaration>} {<form>}**)**
Man kann nun leicht ein Makro Numeric-if definieren, das einen numerischen Wert auf gleich, größer oder kleiner Null abtestet und in Abhängigkeit vom Ausgang des Tests entweder die Zf-, Gf- oder die Lf-Form auswertet (Bild 28).

```
(defmacro numeric-if (nv zf gf lf)
  (let ((sym (gensym)))
       '(let ((,sym ,nv))
             (cond ((zerop ,sym) ,zf)
                   ((> ,sym 0) ,gf)
                   ((< ,sym 0) ,lf)))))
```

Bild 28

Dieses Beispiel demonstriert einen wichtigen Sachverhalt. Wäre der Wert der Auswertung des Arguments nv nicht gebunden, so würde nv in der erzeugten Form dreimal enthalten sein und je nach seinem Wert ein- bis dreimal ausgewertet werden, was – abgesehen vom Zeitverhalten – insbesondere bei Funktionen mit Seiteneffekten zu Fehlern führen kann. Deshalb wird dieser Wert an ein Symbol gebunden, das vorher von der Funktion Gensym bereitgestellt wurde. Die Funktion Gensym erzeugt ein im Lisp-System noch nicht vorhandenes Symbol, das jedoch nicht in die Namenstabelle des Systems eingetragen wird.
Da in Common Lisp If die Basisform ist, kann man Cond durch If beschreiben (Bild 29).

```
(defmacro mp-cond (&rest clauses)
 (let ((test (caar clauses))
       (is-body (cdar clauses))
       (body (if (cddar clauses)
                 '(progn ,@(cdar clauses))
                 (cadar clauses))))
    (if (null clauses)
        nil
        (if is-body
            (if (eq test t)
                body
                '(if ,test ,body
                     (mp-cond ,@(cdr clauses))))
            (let ((sym (gensym)))
             '(let ((,sym ,test))
                (if ,sym ,sym
                    (mp-cond ,@(cdr clauses)))))))))
```

Bild 29

Hierbei ist zu beachten, daß der Körper einer Klausel mehrere Formen enthalten kann, die dann in ein Progn eingekleidet werden müssen, daß eine Klausel keinen Körper haben muß, so daß der Test im positiven Fall auch das Ergebnis liefert, und daß die letzte Klausel den Test T haben kann, so daß keine If-Form mehr erzeugt werden muß.

Das verallgemeinerte Printmakro

Noch einmal zurück zur Ausgabe. In der Literatur wird manchmal ein Printmakro vorgeschlagen, das ähnlich wie Format eine formatierte Ausgabe ermöglicht und vom Nutzer erweitert werden kann. So soll beispielsweise die Anweisung
(let ((a10) (b 20))
(msg nil t
"Der Flächeninhalt des Feldes"
a " x " b
"ist gleich" 5 (∗ a b) "." t))
folgende Ausgabe erzeugen:
Der Flächeninhalt des Feldes 10 × 20
ist gleich 200.
Das angegebene T wird in einen Terpi-Aufruf umgewandelt, positive Zahlen bewirken die Ausgabe von Leerzeichen, Zeichenketten werden unverändert und andere Lispausdrücke nach erfolgter Auswertung ausgegeben (Bild 30).
Hier werden sowohl Zahlen als auch Schlüsselwörter im Car von Listen zur Steuerung der Formatierung benutzt. Man kann dieses Makro erweiterbar machen, indem man zwei Funktionen Set-msg-method und Get-msg-method definiert. Set-msg-method legt in der P-Liste des Symbols MSG unter einem Indikator, dem Schlüsselwort, die zugehörige Funktion ab. Get-msg-method stellt diese Funktion wieder zur Verfügung. Man könnte diese Informationspaare auch in A-Listen (diese werden im nächsten Teil behandelt) oder in Hashtabellen (die leider nicht in allen

```
(defmacro msg (stream &rest args)
 (let ((sym (gensym)))
  `(let ((,sym ,stream))
    ,@(do* ((argl args (cdr argl))
            (arg (car argl) (car argl))
            (res nil))
           ((null argl) (nreverse res))
        (setq res
         (cons
          (cond ((eq arg 't) `(terpri ,sym))
                ((stringp arg) `(princ ,arg ,sym))
                ((numberp arg)
                 (numeric-if arg
                    `(fresh-line ,sym)
                    `(write-spaces ,sym ,arg)
                    `(write-newlines ,sym
                                     ,(- arg))))
                ((atom arg) `(prin1 ,arg ,sym))
                (t (case (car arg)
                    (:eval (cadr arg))
                    (:lines
                     `(write-newlines ,sym
                                      ,(cadr arg)))
                    (:spaces
                     `(write-spaces ,sym
                                    ,(cadr arg)))
                    (:pp
                     `(pprint ,(cadr arg) ,sym))
                    (otherwise
                     `(prin1 ,arg ,sym)))))
              res))))))

(defun write-spaces (stream count)
     (dotimes (k count) (write-char #\Space)))

(defun write-newlines (stream count)
     (dotimes (k count) (terpri stream)))
```

Bild 30

Dialekten enthalten sind) ablegen. Die Case-Anweisung im T-Zweig des Cond müßte dann wie folgt verändert werden:

(t (let ((fn (get-msg-method (car arg))))
(if fn (apply fn ,sym (cdr arg))
'(prin1 ,arg ,sym))))

Der Terminkalender

Im folgenden möchte ich Ihnen einige mit Hilfe dieses Makros geschriebene Ausgabefunktionen für das Programm des Terminkalenders vorstellen.

Die erste Funktion schreibt eine Terminliste auf den Stream ∗STANDARD-OUTPUT∗, das heißt auf das Terminal (wenn ∗STANDARD-OUTPUT∗ nicht anders gebunden wurde). Die benutzte Funktion, die einen Termin ausgibt, wird wieder auf Funktionen zurückgeführt, die die Zeit bzw. den Inhalt des Termines ausgeben (Bild 31). An diesem Beispiel sieht man einen wesentlichen Vorteil des MSG-Makros gegenüber Format: Die Ausgabefolge kann eigene Printfunktionen enthalten, im Gegensatz zu Format, das eine Funktion (und kein Makro) ist und somit keine verzögerte Auswertung ermöglicht.

Aufgabe 9:
Schreiben Sie für das MSG-Makro eine Methode :when mit zwei Argumenten. Das erste Argument ist ein Test, der ausgewertet wird, das zweite Argument soll aber nur ausgewertet werden, wenn der Test wahr ergeben hat.

Schreiben Sie anschließend damit die Funktion Write-time, die die Uhrzeit im Format "hh.mm Uhr" (wenn notwendig mit führender Null) ausgibt und als Ergebnis liefert.

Die nächste Funktion wird nur der besseren Lesbarkeit des Programmes wegen definiert.

(defun write-content (content) (princ content))

Zum Eintragen bestehender Termine in den Kalender gibt es die Funktion Update-calendar. Der Planung eines gemeinsamen Termins zweier Personen dient die Funktion Plan-ap (Bild 32).

```
(defun write-apl (apl)
     (mapc #'write-ap apl) apl)

(defun write-ap (ap)
     (msg nil t "von " (:eval (write-time (ap-start ap)))
                "bis " (:eval (write-time (ap-stop ap)))
                1 (:eval (write-content (ap-content ap))))
     ap)
```

Bild 31

```
(defun update-calendar (person)
  (loop (terpri)
        (princ "Datum [dd mmm]: ")
        (new-ap person (read) (read) (read-ap))
        (princ "noch mehr Eintäge ? [j/n] ")
        (if (not (eq (read) 'j)) (return nil))))

(defun read-date ()
     (princ "Datum [dd mmm]: ")
     (make-date :day (read) :mon (read)))

(defun read-ap ()
     (princ "von [hh mm]: ")
     (let ((start (read-time)))
          (princ "bis [hh mm]: ")
          (make-ap :start start :stop (read-time)
                   :content (read-content))))

(defun read-time ()
     (make-time :h (read) :m (read)))

(defun read-content ()
     (princ "Ereignis [mit RETURN abschließen]: ")
     (read-line))

(defun plan-ap ()
   (common-ap
      (progn (terpri) (princ "Person1: ") (read))
      (progn (terpri) (princ "Person2: ") (read))
      (read-date)
      (progn (terpri)
             (princ "Zeitdauer (min): ")
             (read))
      (read-content)))

(defun common-ap (p1 p2 date dur content)
 (let* ((day (date-day date))
        (mon (date-mon date))
        (time (common-time p1 p1 day mon dur
                 *arbeitsbeginn* *arbeitsende*)))
      (if time
          (let* ((first (car time))
                 (start (ap-start first))
                 (ap (if (= (duration first) dur)
                         first
                         (make-ap :start start
                            :stop
                            (compute-end start
                                         dur)
                            :content content))))
             (new-ap p1 day mon ap)
             (new-ap p2 day mon ap))
          nil)))
```

Bild 32

Aufgabe 10:
Schreiben Sie die (iterative) Funktion Compute-end, die die Zeitstruktur Time und die Zeitdauer in Minuten bekommt und die um diesen Zeitraum verschobene Zeit berechnet. Die Funktion New-ap mit den Argumenten Person, Tag, Monat und Termin, die den neuen Termin in den Kalender der Person eintragen soll, können Sie mit der Funktion Find-day-entry und der Funktion Insert-ap schreiben. Die Funktion Insert-ap soll die Argumente Termin und Terminliste bekommen und eine neue Terminliste mit dem richtig einsortierten Termin erzeugen. Mit der Funktion Save-calendar kann man die Kalender einer Liste von Personen auf ein File schreiben, das man später mit Load wieder einlesen kann, um mit den gesicherten Daten weiterarbeiten zu können (Bild 33).

wird fortgesetzt

```
(defun save-calendar (filename &rest persons)
 (let ((*standard-output*
         (open filename :direction :output)))
      (mapc #'save-calendar1 persons)
      (close *standard-output*)))

(defun save-calendar1 (person)
     (terpri)
     (pprint `(setf (get ',person 'calendar)
                    ',(get person 'calendar)))
     (terpri))
```

Bild 33

```
Aufgabe 9:
(:when `(if ,(cadr arg) ,(caddr arg)))

Die Funktion Write-time kann so aussehen:
(defun write-time (time)
 (msg nil (:when (< (time-h time) 10) (princ 0))
          (:eval (princ (time-h time)))
          (:when (< (time-m time) 10) (princ 0))
          (:eval (princ (time-m time)))  " Uhr")
 time)

Aufgabe 10:
(defun compute-end (time dur)
 (do* ((h (time-h time) (1+ h))
    (m (time-m time) 0)
    (du dur (- du rest))
    (rest (- 60 m) (- 60 m)))
   ((< du rest) (make-time :h h :m (+ m du)))))

(defun new-ap (person day mon ap)
 (let ((entry
         (find-day-entry day mon
         (get person 'calendar))))
       (setf (day-apl entry)
         (insert-ap ap (day-apl entry)))
       entry))

(defun insert-ap (ap apl)
 (cond ((null apl) (list ap))
     ((t< (ap-start (car apl))
          (ap-start ap))
      (cons (car apl) (insert-ap ap (cdr apl))))
     (t (cons ap apl))))
```

Bild 34 Lösungen der Aufgaben 9 und 10

```
Aufgabe 4:
(defun mp-length (li)
 (cond ;;Abbruchtest
       ((null li) 0)
       ;;Rekursion
       (t (+ 1 (mp-length (cdr li))))))

Aufgabe 5:
(defun mp-reverse(li)
  (mp-reverse-aux li nil))

(defun mp-reverse-aux (li acc)
  (cond ;;Abbruchtest
        ((null li) acc)
        ;;Restrekursion
        (t (mp-reverse-aux (cdr li)
                 (cons (car li) acc)))))

Aufgabe 6:
(defun mp-reverse(li)
 (declare (special li))
 (let (res)
      (declare (special res))
      (mp-reverse-aux li)
      res))

(defun mp-reverse-aux(l)
 (when l ;;Rückwärtsrekursion
         (mp-reverse-aux (cdr l))
         (setq res (cons (car li) res)
               li (cdr li))))

Aufgabe 7:
(defun find-day-entry (day month cal)
 (cond ;;Abbruch
       ((null cal) nil)
       ;;Definition lokaler Variable
       ;;für Rekursion
       (t (let* ((entry (car cal))
                 (date (day-date entry)))
               (cond ;;Datum gefunden ?
                     ((and
                        (eq (date-mon date)
                             month)
                        (= (date-day date)
                             day))
                      entry)
                     ;;Rekursion
                     (t (find-day-entry day
                           month
                           (cdr cal))))))))

Aufgabe 8:
(defun t>= (t1 t2)
 (or (> (time-h t1) (time-h t2))
     (and (= (time-h t1) (time-h t2))
          (>= (time-m t1) (time-m t2)))))

(defun t< (t1 t2)
 (or (< (time-h t1) (time-h t2))
     (and (= (time-h t1) (time-h t2))
          (< (time-m t1) (time-m t2)))))

Die Funktion Duration:
(defun duration (ap)
 (+ (* 60 (- (time-h (ap-stop ap))
             (time-h (ap-start ap))))
    (- (time-m (ap-stop ap))
       (time-m (ap-start ap)))))
```

Bild 35 Lösungen der Aufgaben 4 bis 8 aus dem Teil 2

Die Nantucket Tools II

Uwe Holz, Berlin

Zum Softwareentwicklungssystem Clipper (vgl. auch MP 3/90) wird von der Nantucket GmbH Leverkusen eine Sammlung von etwa 500 Funktionen angeboten, die die Möglichkeiten sowohl der Version Sommer 87 als auch der neuen Version 5.0 beträchtlich erweitern. Diese Toolbox einschließlich der Referenzkarte auf den beiden inneren Umschlagseiten vorzustellen, soll Anliegen des folgenden Beitrags sein.

Bei der Entwicklung der Nantucket Tools II wurde die Eigenschaft der Clippersprache, nutzerdefinierte Funktionen (User Defined Functions oder UDFs) definieren zu können, ausgenutzt. Mit den ebenfalls zur Verfügung gestellten komfortablen Schnittstellen für die Sprachen Assembler und C ist die Entwicklung solcher UDFs nicht nur auf die Clippersprache beschränkt. Auch die Einbindung von Modulen anderer Programmiersprachen wie Pascal ist möglich, allerdings müßte sich in diesem Falle der Programmierer selbst um die Parameterübergabe/-übernahme und Werterückgabe kümmern. Einzige Voraussetzung ist ein zu Microsoft kompatibles Format der Objektdateien. Diese offene Architektur ist Voraussetzung für die Schaffung zahlreicher Zusatzbibliotheken und Toolboxen, die Clipper für den Programmierer qualitativ wesentlich aufwerten.
Die Toolbox bietet viele Funktionen, die in der praktischen Arbeit des Anwendungsprogrammierers häufig gebraucht werden, so daß dadurch die Erstellung von UDFs auf ein Minimum reduziert wird.
Aufbauend auf den Nantucket Tools I wurden alle Themenkomplexe konsequent erweitert, wobei die mathematischen und die Netzwerkfunktionen, Funktionen zum Lesen und Schreiben von Speicherinhalten bzw. Ports sowie zur Unterstützung der Programmierung der Tastatureingabe (erweiterte Get-Funktionen) neu entstanden.
Die Toolbox ist vollständig in Assembler programmiert, zeichnet sich daher durch geringen Speicherbedarf und geringe Laufzeiten aus, und wird an neue Clipper-Versionen angepaßt, wobei bisherige Funktionen auch weiterhin nutzbar sind. Dabei wurden sowohl vorhandene Funktionsgruppen als auch der Sprachstandard von Clipper insgesamt erweitert. Die Bibliothek ist modular aufgebaut, so daß nicht benötigte Funktionsgruppen die Anwendung nicht unnötig belasten.
Eine Einschränkung hinsichtlich des Einsatzes der Nantucket Tools II soll jedoch an dieser Stelle nicht unerwähnt bleiben: Der Bereich der grafischen Funktionen (Geschäftsgrafik, Präsentation) wird nicht unterstützt. Nutzer, die derartige Funktionen verwenden wollen, müssen auf andere Tools zurückgreifen (Flipper, Dgraph, KRS-Grafik-Toolbox etc.).
Von besonderem Interesse – vor allem auch im Hinblick auf die erforderliche anwenderfreundliche Programmgestaltung – dürften die Funktionen sein, die die Fenstertechnik ermöglichen bzw. deren Programmierung erleichtern. Wollen Sie die zahlreichen zur Verfügung stehenden *Fensterfunktionen* nutzen, so ist es außerdem erforderlich, den Treiber NTOOLD87.OBJ mit einzubinden, und zwar so, daß er den Standardbildschirmtreiber der CLIPPER.LIB für diese Funktionen überschreibt:
Fl ihr-prog, NTOOLD87
Während einer Anwendung können dann bis zu 255 Fenster eingerichtet werden, vorausgesetzt, es ist genügend Hauptspeicherkapazität vorhanden. Nach dem Öffnen der einzelnen Fenster (siehe auch Bild 1) können diese dann unter Angabe der Fensternummer aktiviert und auch wieder geschlossen werden.
Der bereits erwähnte *erweiterte Treiber NTOOLD87.OBJ* ist aber nicht nur für die Fensterfunktionen, sondern auch beim Einsatz einiger weiterer selbständiger Funktionen erforderlich. Er gewährleistet, daß die Interrupt-Vektoren des Betriebssystems beim Start der Anwendung automatisch gesichert und nach Beendigung (Return, Quit) wiederhergestellt werden. Auch die Kursorformen und -farben werden auf die Werte vor dem Start der Anwendung zurückgesetzt.
Bestandteil des Treibers sind weiterhin Funktionen, die Druck- und Tastaturübersetzungstabellen installieren oder andere, sonst unter Clipper nicht zu beeinflussende Einstellungen modifizieren. Nach Einbindung des Treibers in die Anwendung wird dieser als zentrales Ein-/Ausgabemodul benutzt und hat damit auch Einfluß auf eine Reihe von Bildschirmfunktionen.
Die Gruppe der *Schnittstellenfunktionen* ermöglicht es, bis zu vier serielle Schnittstellen gleichzeitig anzusteuern. Es handelt sich hierbei um 38 völlig neue Funktionen, die die Übertragung von Daten über serielle Schnittstelle gewährleisten. Für jeden Port kann ein Sende- und Empfangspuffer mit einer Größe von bis zu 64 KByte angelegt werden. Sowohl das Senden als auch das Empfangen sind im Hintergrund möglich. Alle Schnittstellenparameter (Übertragungsrate, Parität, Datenlänge, Stopbits) sind frei modifizierbar, ohne die entsprechende Schnittstelle schließen zu müssen, das heißt, die Pufferinhalte bleiben erhalten. Wichtige Steuersignale wie CD (Carrier Detect), DTR (Data Terminal Ready), CRC (Cyclic Redundancy Check) usw. können abgefragt und gesetzt werden. Voraussetzung für das einwandfreie Funktionieren ist allerdings eine völlige Kompatibilität zum IBM PC, da die Hardware unter Umgehung der BIOS- und DOS-Aufrufe direkt angesprochen wird. Die Datenübertragung im Hintergrund ist aber bei gleichzeitiger direkter Ausgabe auf den Bildschirm mit der Version 1.0 der Toolbox problematisch.
Die Gruppe von Funktionen beinhaltet nahezu alle für Zeichenketten möglichen Operationen. Die *Zeichenkettenfunktionen* gestatten die Programmierung komfortabler Texteditoren und können somit auch zur Bearbeitung von MEMO-Feldern genutzt werden. Es gelten auch hier die durch Clipper vereinbarten Konventionen hinsichtlich der maximalen Länge von 64 KByte. Dabei kann es jedoch leicht zu Speicherplatzproblemen kommen, vor allem wenn solche Zeichenketten als Parameter bei Funktionsaufrufen verwendet werden. Hier ist die Parameterübergabe per Reference zu empfehlen, bei der statt des Parameters lediglich ein Adreßzeiger auf den Originalstring übergeben wird. Diese Methode spart Zeit und vor allem Speicherplatz, da sonst kurzzeitig drei Kopien der Variablen im Hauptspeicher existieren, sie kann lediglich nicht bei Funktionen verwendet werden, die die Länge eines Strings verändern.

Beispiel:
Übergabe per Wert
v1 = "STRING"
v1 = MY_FUNC(v1)
Übergabe per Reference
v1 = MY_FUNC(@v1)

Zu der Gruppe der Funktionen für die *Zahlenumwandlung und die Bitmanipulationen* gehören vor allem solche zur Konvertierung zwischen den einzelnen Zahlensystemen, zur Erzeugung von Zufallszahlen sowie zur bitweisen Manipulation von Zahlen und Zeichenketten entsprechend den üblichen Binäroperationen wie AND, XOR, OR, NOT oder das Prüfen von Bits. Beispielsweise werden Funktionen wie NUMLOW() und NUMHIGH() im Zusammenhang mit Tollboxfunktionen benötigt, die zwei 8-Bit-Werte als kombinierte 16-Bit-Zahl, und damit quasi zwei Ergebnisse mit einem Return-Wert zurückgeben.
Der Bereich der *Bildschirmfunktionen* gestattet die zeilen- und spaltenweise Bildschirmmanipulation, den Test des angeschlossenen Bildschirmadapters, das Speichern und Rückspeichern von Bildschirminhalten und die Manipulation von Bildschirm-, Kursor- und Farbattributen. Sie harmonieren eng mit den Teiber- und Fensterfunktionen, so daß sich bestimmte Einstellungen und Wirkungen von Funktionen dieser 3 Gruppen gegenseitig beeinflussen.
Vor allem das präventive Erkennen von Fehlern, die entweder durch den Nutzer oder durch die Projektumgebung bzw. das Betriebssystem-Environment während der Programmlaufzeit verursacht werden könnten, ermöglichen die *Disketten- und Festplattenfunktionen*. Um diese Fehler möglichst abzufangen, stehen Funktionen zum Ermitteln der Größe der Dateien und der freien Kapazität auf Datenträgern, zum Ändern der Dateiattribute und zum Ermitteln der Laufwerksbereitschaft sowie normale Betriebssystemfunktionen zur Datei- und Verzeichnismanipulierung (Löschen, Kopieren, Verschieben usw.) zur Verfügung.
Auch die *Druckerfunktionen* dienen vor allem dazu, mögliche Bedienfehler auszuschließen bzw. ihnen zuvorzukommen und so beispielsweise die Druckerbereitschaft zu testen. Außerdem ermöglichen sie es auch, angeschlossene Drucker zu initialisieren, Schriftfonts einzustellen und Druckdateien an die Warteschlange des Netzes zu übergeben, um dadurch Wartezeiten vermeiden zu können.
Im dBase-Sprachstandard existieren bereits Funktionen, die mit dem Variablentyp Datum rechnen. Für bestimmte finanztechnische Berechnungen reicht dies jedoch nicht aus und so werden zahlreiche *Datums- und Zeitfunktionen* durch die Toolbox bereitgestellt.
Einige sinnvolle Erweiterungen der *Datenbankfunktionen* geben weitere Auskünfte über die Größe der DBF-Datei, über die Da-

```
* --------------------------------------------------------------------------
*| MakeWin()      : Fenster explosionsartig öffnen mit Schatten              |
*| Autor          : Uwe Holz                                                 |
*| Datum          : 08.05.90                                                 |
*| Parameter      : z_oben      - obere Zeile    (Typ N)                     |
*|                  sp_links    - linke Spalte   (Typ N)                     |
*|                  z_unten     - untere Zeile   (Typ N)                     |
*|                  sp_rechts   - rechte Spalte  (Typ N)                     |
*|                  header      - Titelzeile (optional)                      |
*| Rückgabe       : Typ N - Fensternummer                                    |
*| Aufrufe        : GETBOXGROW(), SETBOXGROW(), COLORWIN(),                  |
*|                  WOPEN(), WFORMAT(), WBOX(), SAYSPREAD()                  |
*| Beispiel       : MakeWin(3,2,22,76,"< H i l f e >")                       |
* --------------------------------------------------------------------------
FUNCTION MakeWin
PARAMETERS z_oben, sp_links, z_unten, sp_rechts, header
PRIVATE cur_atr, cur_grow
PRIVATE tmp_n

cur_grow = GETBOXGROW()  && derzeitige Verzögerungseinstellung
                         && sichern
SETBOXGROW(5)            && Verzögerung einstellen
IF (z_oben<0 .OR. z_unten>23 .OR. sp_links<0 .OR. sp_rechts>77)
  tmp_n = -1
ELSE
  tmp_n = WOPEN(z_oben, sp_links, z_unten+1, sp_rechts+2)
  WFORMAT(0,0, 1, 2)       && Fenster öffnen
  WBOX(1)
  WFORMAT(-1,-1,-2,-3)
  COLORWIN(z_unten-z_oben+1,2, z_unten-z_oben+1,;
           sp_rechts+2-sp_links)
  COLORWIN(1, sp_rechts-sp_links+1, z_unten-z_oben+1,;
           sp_rechts-sp_links+2)
  WFORMAT(0,1,2,3)
  IF PCOUNT() < 5
  ELSE
     SAYSPREAD(header,50,0)
  ENDIF
  WFORMAT(1, 0, 0, 0)
ENDIF
SETBOXGROW(cur_grow)     && alten Verzögerungswert zurückstellen
RETURN tmp_n
```

***Bild 1** Fenster werden explosionsartig mit Schatten geöffnet.*

```
* --------------------------------------------------------------------------
*| Srch_Rec()     : Zeichenkette in einer DBF-Datei im gesamten            |
*|                  Datensatz suchen                                       |
*| Autor          : Uwe Holz                                               |
*| Datum          : 16.07.90                                               |
*| Parameter      : rec_nr   - Satz, bei dem begonnen wird (Typ)           |
*|                  suchwort - zu suchende Zeichenkette    (Typ)           |
*| Rückgabe       : Typ N - Satznummer                                     |
*|                        > 0 (gefunden)                                   |
*|                        = 0 (nichts gefunden)                            |
*|                        < -1 (keine Datenbank in USE)                    |
*| Aufrufe        : ALLOFREE(), _DBFHANDLE(), ATNUM()                      |
*| Voraussetzung: Eine Datenbank ist eröffnet.                             |
*| Datenbank      : Name der zu bearbeitenden Datenbank                    |
*| Beispiel       : rec = Srch_Rec(1,"Clipper")                            |
* --------------------------------------------------------------------------

FUNCTION Srch_Rec
PARAMETERS rec_nr, suchwort
PRIVATE puffer, num, pos, anz, dbf_hdle, ret, hdr, akt_rec
gef = .F.
akt_rec = RECNO()                    && den aktuellen Satz,
recs = RECSIZE()                     && die Satzlänge und
hdr = HEADER()                       && die Headergröße ermitteln
dbf_hdle = _DBFHANDLE()              && Nummer des Zugriffsmodus
IF dbf_hdle > 0                      && >0, dann ist die Datei eröffnet
   num = ALLOFREE()                  && freier Speicher
   IF num > 65535
      num = 65535                    && maximale Größe
   ENDIF
   num = INT(num/recs) * recs        && Puffergröße auf Vielfaches
   puffer = SPACE(num)               && der Satzgröße
   anz = num
   ret = 0
   DO WHILE anz = num
      FSEEK(dbf_hdle,(rec_nr-1)*recs + hdr,0)  && positionieren
      anz = FREAD(dbf_hdle, @puffer, num)      && lesen
      pos = ATNUM(suchwort,@puffer,1)          && suchen
      IF pos > 0
         ret = rec_nr + INT(pos/recs)          && gefunden
         EXIT
      ELSE
         rec_nr = rec_nr + INT(anz/recs)       && nicht gefunden,
      ENDIF                                    && weitersuchen
   ENDDO
ELSE
   ret = -1                 && keine Datei eröffnet
ENDIF
IF ret > 0
   GOTO ret                 && Suchwort gefunden, Satz einstellen
ELSE
   GOTO akt_rec             && nicht geunden, alten Satz einstellen
ENDIF
RETURN(ret)
```

***Bild 2** In einer Datenbank wird eine Zeichenkette im gesamten Datensatz gesucht.*

tenbankfelder oder testen auf das Vorhandensein einer DBT-Datei, falls MEMO-Felder in der Datenbank enthalten sind. Das Bild 2 zeigt die Suche einer Zeichenkette in allen Feldern eines Datensatzes.
Sämtliche aktuellen Set-Einstellungen sowie das Vorhandensein des Debuggers können mit speziellen *Statusfunktionen* ermittelt werden.
Wichtige Systeminformationen über das Betriebssystem (DOS) oder die Hardware (BIOS) können Sie über die sogenannten *Systemfunktionen* erhalten, die auch die Anpassung der Anwendung an die konkrete Hard- und Softwareumgebung zum Zeitpunkt des Programmlaufes gestatten.
Für MS-DOS ab der Version 3.1 und die Netzwerke PC-LAN, MS-NET und Novell Netware bietet die Toolbox zahlreiche *Netzwerkfunktionen*, die ausschließlich auf dem Novell Application Programmers Interface (API) basieren und beispielsweise Informationen über den Server und das Printer-Management liefern.
Zum Lieferumfang der Nantucket Tools II gehören außerdem noch einige eigenständige Programme, die jedoch nicht in der Referenzkarte aufgeführt sind. Dazu gehören Beispielprogramme für die Toolbox (Tooldemo.prg) und auch der Clipper-Quellkode für einen Font-Editor für VGA-/EGA-Fonts (Fontedit.prg) und für ein Programm zum Einstellen der Farbpaletten über „Schieberegler" (Colorpal.prg). Zwei weitere Programme dienen dem Speichern und Restaurieren von Interrupt-Vektoren (Intsave.com) bzw. als „Notbremse" für geänderte Interrupts (Intoff-.com).

✉ KONTAKT ☏

risc Hard- & Software GmbH, Kreuziger Str. 23, O-1035 Berlin

Schlecht beraten

Die Steuerpflicht des selbständigen DV-Beraters

Dr. Franz Otto

Freiberufler haben dem Finanzamt gegenüber einen besseren Stand: Sie sind nicht gewerbesteuerpflichtig. Selbständige DV-Berater möchten sich diesen Umstand gern zunutze machen. Ein aktuelles Urteil des Bundesfinanzhofes macht jedoch viele Hoffnungen zunichte.

Der Bundesfinanzhof hat die Frage erörtert, ob ein selbständiger DV-Berater, der Anwendungsprogramme entwickelt, gewerbesteuerpflichtig ist (Urteil vom 7. 12. 89 – IV R 115/87). Der Berater selbst meinte, er übe einen dem Ingenieur ähnlichen Beruf aus und unterliege dieser Steuer deshalb nicht, weil Ingenieure und ähnliche Berufe den Status des Freiberuflers innehätten.
Unstreitig war, daß der Berater nicht den Beruf eines Ingenieurs ausgeübt hatte. Weil er die vorgeschriebene Ausbildung nicht absolviert hatte, durfte er die Berufsbezeichnung *Ingenieur* nicht führen. So kam es darauf an festzustellen, ob der Berater eine dem Ingenieurberuf „ähnliche" Tätigkeit ausgeübt hatte. Diese liegt vor, wenn sie in ihren wesentlichen Punkten mit einem der im Einkommensteuergesetz (§ 18) genannten Berufe verglichen werden kann.
Dazu ist zum einen erforderlich, daß die Tätigkeit in einem für den Beruf des Ingenieurs typischen Bereich gelegen hat. Aufgabe des Ingenieurs ist es, auf der Grundlage natur- und technikwissenschaftlicher Erkenntnisse und unter Berücksichtigung wirtschaftlicher Belange technische Werke zu planen, zu konstruieren und ihre Fertigung zu überwachen.
So ist der Bundesfinanzhof bereits 1983 zu dem Ergebnis gekommen, daß ein selbständiger Diplominformatiker (dessen Ausbildung der der Ingenieure vergleichbar ist) eine dem Ingenieurberuf ähnliche Tätigkeit ausübt, wenn er Systemanalysen erarbeitet, aus denen sich ergibt, daß gewisse betriebliche Vorgänge mit Hilfe von DV-Anlagen nachvollziehbar sind.

So nicht

Dies will der Bundesfinanzhof jetzt nicht mehr uneingeschränkt so gelten lassen. Er hält allerdings daran fest, daß der Diplominformatiker mit Hochschul- oder Fachhochschulabschluß von seiner Ausbildung her einem Ingenieur vergleichbar ist: Der Studiengang

des Diplominformatikers ist weitgehend von der Mathematik, aber auch von Fächern wie Nachrichten- und Elektrotechnik geprägt.
Die Ausbildung zum Diplominformatiker befähigt demnach dazu, auf der Grundlage natur- und technikwissenschaftlicher Erkenntnisse Werke zu planen, zu konstruieren und ihre Fertigung zu überwachen – mögen die „Werke" auch nicht immer als körperliche Gegenstände (Hardware) in Erscheinung treten, sondern als Software. Diese Einordnung wird dadurch bestätigt, daß an manchen Fachhochschulen der Studiengang Informatik zum Ingenieurabschluß führt.
Der Bundesfinanzhof hält nach erneuter Prüfung nicht daran fest, daß ein Diplominformatiker immer dann, wenn er Systemanalysen ausführt, einer Tätigkeit nachgeht, die in einem ingenieurtypischen Bereich liegt. Der Begriff Systemanalyse sei zu weit gefaßt und zu ungenau, als daß er zur Abrenzung herangezogen werden könnte.
Die Richter fordern daher eine Präzisierung, was zu einer Einschränkung der ingenieurähnlichen Tätigkeiten im Bereich der Datenverarbeitung führt. Unter Systemanalyse versteht man zum einen die Tätigkeit des Systemsoftware-Entwicklers, zum anderen aber überwiegend die Anwendersoftware-Entwicklung – in Gesetzen und Tarifverträgen als *Datenverarbeitungsorganisation* bezeichnet.
Das typische Berufsbild für den an einer Fachhochschule oder wissenschaftlichen Hochschule ausgebildeten Diplominformatiker, den sogenannten Kerninformatiker, ist – jedenfalls im Software-Bereich – nur das der Systemsoftware-Entwicklung. Dazu gehört zunächst die Grundlagenforschung, das heißt die Systematisierung und Gestaltung der den Rechenanlagen eigenen Strukturen. Dabei geht es um die allgemeinen Gesetze, die der Informationsverarbeitung zugrunde liegen: um Algorithmen, künstliche, formale Sprachen und deren Syntax und Semantik.

Voraussetzungen

Quantitativ gewichtiger zeigen sich die Aufgaben bei der Planung von System-Software. Es geht dabei vor allem um den Entwurf von Strukturen zwischen der Hardware und der Anwendernahtstelle. Dies sind vor allem Betriebssysteme, also die Systeme, ohne die Computer nicht einsatzfähig sind: Hilfs- und Dienstprogramme, Compiler und Übersetzer oder Datenbanksysteme.
Solche Systeme entwirft der Systemsoftware-Entwickler, fixiert und detailliert sie im Pflichtenheft. Die Vielfalt und Komplexität bereits bestehender Programme, die Varianten bei der Hardware-Ausstattung, verbunden mit der Forderung nach störungsfreiem, schnellem Ablauf, kennzeichnen die Ansprüche, die an die Ausbildung in diesem Bereich gestellt werden. Die Tätigkeit setzt eigentlich stets eine an Hoch- oder Fachhochschulen erworbene Informatikausbildung voraus.
Demgegenüber ist die Anwendersoftware-Entwicklung, obwohl sie in vielen Fällen nicht weniger anspruchsvoll sein mag als Systemsoftware-Entwicklung, keine ingenieurtypische Tätigkeit. Dies ist die entscheidende Wertung des Bundesfinanzhofes.

Differenzen

Der Unterschied zum Systemsoftware-Entwickler besteht darin, daß mindestens gleichberechtigt neben die Informatikkenntnisse Wissen im jeweiligen Anwendungsbereich (sei es Betriebswirtschaft, Technik, Medizin, Recht) tritt.
Abschließend hat der Bundesfinanzhof noch die Frage erörtert, ob der Berater in dem konkreten Fall eventuell eine dem beratenden Betriebswirt ähnliche Tätigkeit ausgeübt hatte – auch das hätte zur Gewerbesteuerfreiheit geführt.
Eine solche Tätigkeit übt aus, wer Kenntnisse in den hauptsächlichen Bereichen der Betriebswirtschaftslehre in einer Breite besitzt, wie sie die Ausbildungsgänge zum Diplomkaufmann, zum Diplombetriebswirt oder zum staatlich geprüften Betriebswirt vermitteln. So ausgestattet muß eine Beratungstätigkeit in einem der Hauptbereiche der Betriebswirtschaftslehre ausgeübt werden.
Nach ständiger Rechtsprechung des Bundesfinanzhofes trifft das auf den DV-Berater indes nicht zu. Dies wird damit begründet, daß der Berater in DV-Angelegenheiten steuerrechtlich als eigenständiger Beruf anzusehen ist, der sich in seiner totalen Ausrichtung auf Theorie und Technologie der Datenverarbeitung wesentlich von dem beratenden Betriebswirt unterscheidet. Daran hält der Bundesfinanzhof weiter fest.

Nachdruck aus c't–Magazin für Computertechnik, Heft 8/1990

Die Überwindung der RAM-Grenzen unter DOS

Hans-Georg Sanner, Ludwigshafen/Rh.

Der Ursprung: Bank Switching unter CP/M

Der verfügbare Speicherraum eines PCs wird durch den Adreßbus des verwendeten Prozessors bestimmt. Die 8-Bit-Prozessoren 8080 und Z 80 hatten einen 16-Bit-Bus, das ergibt 64 KByte Speicherraum – für damalige Verhältnisse mehr als ausreichend.
Für die notwendige „Ordnung" im Speicherraum ist das Betriebssystem zuständig. Das war damals CP/M von Digital Research (bis Version 2.2). Die Speicheraufteilung zeigt Bild 1. Je nach der (individuellen) Größe des BIOS war der für den Anwender verfügbare Speicher, der TPA (Transient Program Area), danach gute 50 KByte groß. Die Ansprüche an ein Betriebssystem waren noch bescheiden, besonderer Komfort – beispielsweise eine grafische Benutzeroberfläche – wurde nicht verlangt.
Die Anwenderprogramme wurden aber bald länger als es der TPA erlaubte. Die Overlay-Technik brachte eine gewisse Abhilfe. Hier wird ein vom Programmierer bestimmter RAM-Bereich mehrfach ausgenutzt, indem gerade benötigter Programmcode (Unterprogramme) von der Diskette in diesen Bereich geladen wird und dort gegebenenfalls nicht benötigten Code überschreibt. (Verwendet wurde dies zum Beispiel bereits bei den frühen Versionen von Wordstar). Andere Programme aber (z. B. Tabellenkalkulationen) forderten einen größeren TPA, da die Größe der im Speicher ständig verfügbaren (residenten) Datenmengen ständig zunahm. Eine „lineare" Speichervergrößerung war wegen der begrenzten Breite des Adreßbusses (16 Bit) nicht möglich, so entstand das Konzept der „nichtlinearen" Speichervergrößerung.
Das Prinzip zeigt Bild 2. Der Speicher wird in einzelne „Blöcke" aufgeteilt. Zu einem bestimmten Zeitpunkt ist eine bestimmte Gruppe von Blöcken als eine Bank aktiv, das

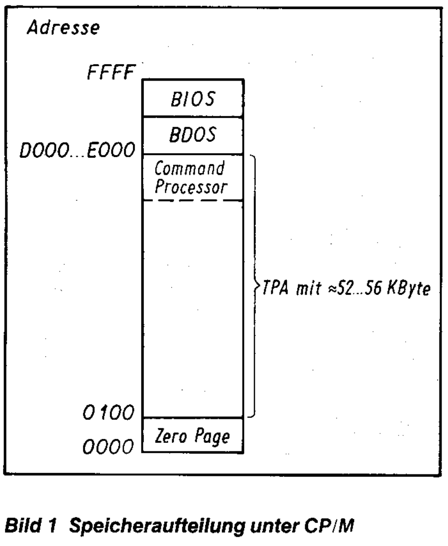


Bild 1 Speicheraufteilung unter CP/M

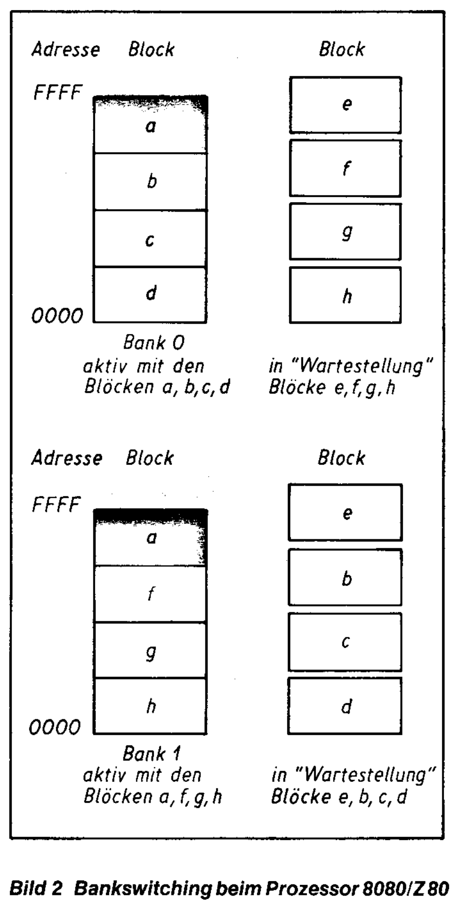


Bild 2 Bankswitching beim Prozessor 8080/Z 80

heißt adressierbar. Über eine entsprechende Hardware-Einrichtung (MMU = Memory Management Unit) können jederzeit ein oder mehrere Blöcke ausgetauscht werden. So entsteht eine andere aktive Bank. Man spricht daher von Bank Switching.
Verwendet wurde das im Betriebssystem CP/M Plus (Bild 3): Ein verkürztes „residentes" BIOS und das BDOS vergrößern den TPA auf etwa 62 KByte in der TPA-Bank. Werden Systemroutinen angesprochen, so wird auf die Systembank umgeschaltet. (In der Systembank können bei Bedarf noch zusätzlich einzelne RAM-Blöcke gegen ROM-Blöcke mit Routinen ausgetauscht werden.)
Bei den heute gebräuchlichen 8-Bit-Rechnern der Firma Amstrad/Schneider (CPC und PCW/Joyce) wird vom Bank Switching reger Gebrauch gemacht. Anwenderprogramme haben es jedoch nicht genutzt, vielleicht, weil das Zeitalter der 16-Bit-Rechner schon begonnen hatte.

Expanded Memory beim PC/XT

Der Prozessor 8086 brachte einen Adreßbus von 20 Bit mit 1 MByte Speicherraum. Als „echter" 16-Bit-Prozessor verfügt er auch über einen Datenbus von 16 Bit. Auf Wunsch von IBM wurde von Intel später noch der Prozessor 8088 als abgemagerte Version mit einem 8-Bit-Datenbus herausgebracht – um Entwicklungskosten für Zusatzhardware zu sparen. Mit diesem Prozessor waren der Original-IBM PC und die meisten Kompatiblen ausgerüstet. Das Betriebssystem MS-DOS (bzw. PC-DOS oder kurz DOS) ergab die Speicheraufteilung in Bild 4. Ein Drittel des Raumes wurde damals großzügig zur Unterstützung des Betriebssystemes reserviert. Aber die verbleibenden 640 KByte schienen wirklich ausreichend genug.
Doch auch hier wurden die Anwenderprogramme immer umfangreicher. Es stiegen die Ansprüche an den Bedienungskomfort – auch beim Betriebssystem – und speicherresidente Programme wurden immer beliebter. Diese TSR-Programme (Terminate and Stay Resident) sollen jederzeit – auch während ein anderes Anwenderprogramm läuft – aufgerufen werden können. Zum Beispiel durch eine bestimmte Tastenkombination (Beispiele sind Sidekick, PC Tools u. a.) oder auch etwa regelmäßig in bestimmten Zeitabständen (Zeitanzeige). Dazu müssen sie ständig resident im Speicher verfügbar sein und außerdem die geeigneten Systeminterrupts „verbiegen" (z. B. Tastatur, Timer usw.).
Als Ergebnis wurde der verbleibende Platz für die aufrufbaren Anwenderprogramme im konventionellen Bereich von 640 KByte immer kleiner. Eine lineare Vergrößerung des Speicherraumes über 1 MByte hinaus war nicht möglich, also besann man sich auf das Prinzip der nichtlinearen Vergrößerung wie bei CP/M und schuf das Konzept von Expanded Memory (erweiterter Speicher, hier im Sinne von *zusätzlich*; Bild 5). Anders als beim CP/M-Vorbild liegen aber hier die umzuschaltenden Blöcke nicht im TPA, sondern in einem Fenster im oberen Adreßbereich (oberhalb von 640 KByte). (Wir werden noch sehen, warum das unter DOS so sein muß.) Die Lage des Fensters wird so gewählt, daß der Platz nicht von System-RAM oder -ROM beansprucht wird (üblicherweise zwischen D0000 und F0000, also „zwischen den Adaptern").
Zur Installation vom Expanded Memory ist wieder sowohl Hard- als auch Software erforderlich. Die Hardwarekomponente besteht aus dem erforderlichen RAM und der Einrichtung zur Einblendung der RAM-Blöcke in das Fenster im oberen Bereich. Die Steuerung erfolgt über einen freien Port des Rechners. Die Software zur Steuerung besteht aus einem Treiber, der vom Programm aus aufgerufen werden kann.
Expanded Memory war von Anfang an für Anwenderprogramme bestimmt (z. B. Tabellenkalkulationen). Damit die Nutzung problemlos erfolgen kann, wurde das Expanded-Memory-Konzept „genormt". Die Firmen Lotus, Intel und Microsoft (LIM) schufen hierzu EMS (Expanded Memory Specification). Gebräuchlich ist zur Zeit noch die Version 3.2 (mit maximal 12 MByte Expanded Memory) neben der neuen Version 4.0 (mit maximal 32 MByte). Die Blockgröße ist immer 16 KByte, die Fenstergröße bei Version 3.2 ist 64 KByte, wobei das Fenster aus 4 hintereinander liegenden Blöcken besteht. Bei EMS 4.0 besteht das Fenster aus maximal 12 Blökken, die aber nicht alle hintereinander liegen müssen (und aus Platzmangel im oberen Bereich auch oft nicht hintereinander liegen können).
Die Hardware befindet sich auf einer Einsteck-Karte. Diese wurde unter der Bezeichnung *Above Board* von Intel herausgebracht. Heute werden solche Karten von den verschiedensten Firmen angeboten. Mit Dip-Schaltern läßt sich die Lage des Fensters einstellen (entsprechend der vorliegenden Hardwarekonzeption des Rechners). Der Softwaretreiber (EMM = Expanded Memory Manager) wird mit einer Device-Anweisung in das System eingebunden (Parameter ist dabei die Lage des Fensters).
Ein Anwendungsprogramm kann der Treiber mit dem DOS-Interrupt 67H direkt ansprechen. Die einzelnen EMS-Funktionen werden durch entsprechende Werte im Register AH aufgerufen /1/, /2/. Nutzung und richtige Handhabung von EMS ist allein Sache des Programmierers /3/, /4/, /5/.
Eine ganze Reihe kommerzieller Programme (vornehmlich Tabellenkalkulationen, wie Lotus 1-2-3 oder Framework) machen regen Gebrauch von EMS. Dabei werden aber üblicherweise nur Daten in das Fenster eingeblendet, obwohl grundsätzlich auch ein (Unter-)Programm in diesem Fenster laufen könnte – wenn es dessen Größe nicht überschreitet. Es ist aber unter DOS nicht so einfach, dort ein Programm zum Laufen zu bringen.
Unter CP/M kann sich zu einem bestimmten Zeitpunkt immer nur *ein* Programm lauffähig im Speicher befinden. Ihm steht während des Ablaufes dann der gesamte TPA für Code und Daten zur Verfügung. Unter DOS ist das grundsätzlich anders. Hier können mehrere Programme gleichzeitig im Speicher sein, von denen jeweils eines läuft, während sich die anderen in „Wartestellung" befinden. DOS verwendet dafür eine besondere Art der Speicherverwaltung. Für jedes Programm wird der (vom Programmierer festgelegte) Speicherplatz direkt im TPA reserviert. Vor jedem solchen Bereich befindet sich ein 16 Byte langer Memory Control Block (MCB), der Angaben über Verwendung und Länge des folgenden Speicherblocks enthält. Außerdem ist dort ein Zeiger auf den nächsten MCB, der sich entsprechend am Anfang des nächsten Speicherblocks befinden muß. Hinter dem letzten belegten Speicherblock beginnt der noch freie RAM – bis zum Ende des TPA (640 KByte). Am Anfang dieses Bereiches ist dann der letzte MCB, der den Bereich als frei kennzeichnet.

Die wichtigsten Begriffe auf einen Blick

Shadow-RAM: RAM-Speicher zwischen 640 KByte und 1 MByte (A000H-FFFFH) bei Boards, die mit Chips&Technologies-Chipsätzen aufgebaut sind. Da RAM schneller als ROM ist, werden der Video-ROM und das System-BIOS in den parallelen RAM kopiert. Ein großer Teil (30 bis 130 KByte) bleibt jedoch ungenutzt.
Extended Memory: linearer Speicher oberhalb 1 MByte, der von manchen Programmen genutzt wird.
Expanded Memory: Speicher nach der LIM-(Lotus-Intel-Microsoft)-Spezifikation. Aktuell ist die Version 4.0. Hierbei wird der Zusatzspeicher oberhalb 1 MByte in entsprechend viele 64-KByte-Seiten geteilt. Der Zugriff erfolgt über ein 64-KByte-Fenster, welches adreßmäßig im Bereich zwischen 640 KByte und 1 MByte liegt.
XMS (Microsoft Extended Memory Specification): Um den Extended Memory besser nutzen zu können, wurde dieser Standard geschaffen. Der dazu passende Treiber von Microsoft nennt sich Himem.sys. XMS teilt sich wiederum in die drei Teilbereiche HMA, UMB und EMB.
HMA (High Memory Area): ist ein 64 KByte großer Block, der direkt oberhalb der 1-MByte-Grenze in 80286- und 80386-Systemen beginnt. Der große Vorteil gegenüber dem restlichen Extended Memory besteht darin, daß in diesen Bereich Programme hineingeladen und ausgeführt werden können. Hierbei handelt es sich also um eine echte Erweiterung des Hauptspeichers, wobei die nutzende Software speziell für den HMA-Zugriff geschrieben sein muß. Dieser Bereich ist nicht zu verwechseln mit dem von Quarterdeck definierten Begriff High-RAM und auch nicht mit dem High Memory.
UMB (Upper Memory Block = High Memory): ist der Adreßbereich zwischen 640 KByte und 1 MByte, der vielfach nutzbar ist. Hier läßt sich Shadow-RAM, Extended oder Expanded Memory einrichten, und es ist eine Erweiterung des Hauptspeichers als High RAM möglich. Das Betriebssystem kann auf diesen Abschnitt genauso wie auf den Hauptspeicher unterhalb 640 KByte direkt zugreifen.
High-RAM: QRAM und QEMM-386 können in den freien Adreßbereich zwischen 640 KByte und 1 MByte Speicher einblenden und diesen dem Betriebssystem als zusätzlichen Hauptspeicher, genannt High-RAM, zur Verfügung stellen. Hier können TSR-Programme ablaufen, ohne den Bereich unterhalb von 640 KByte zu belegen. Als RAM-Quelle für den High-RAM nutzen die beiden Memory Manager entweder aus einem anderen Bereich gespiegelten Expanded Memory (EMS) oder den hardwaremäßig adreßgleichen Shadow-RAM auf NEAT-/C&T-Boards.
TSR (Terminate Stay Resident): Sammelbegriff für speicherresidente Programme, die nach dem Laden im Hauptspeicher verbleiben. Hierzu gehören der Maustreiber, der Tastaturtreiber (KEYBGR o. ä.), aber auch Sidekick uvm.

Nachdruck aus Computer Persönlich, Heft 18/1990

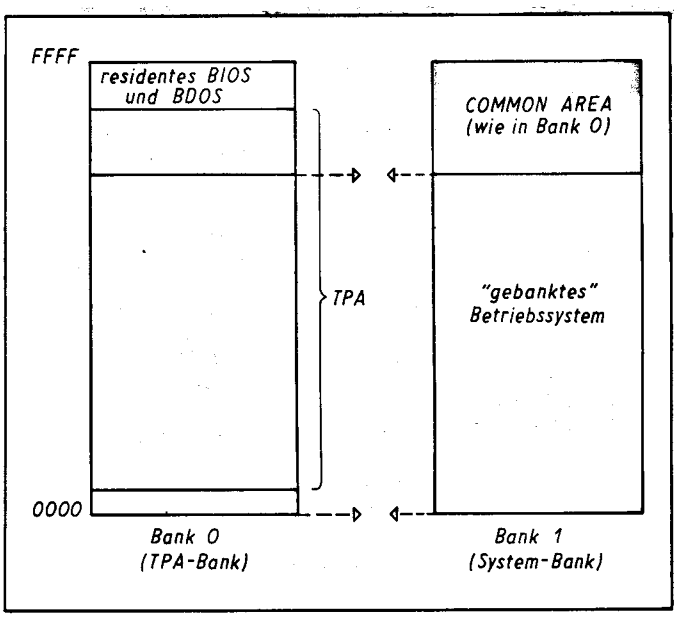


Bild 3 Speicheraufteilung unter CPM/ Plus mit Bankswitching

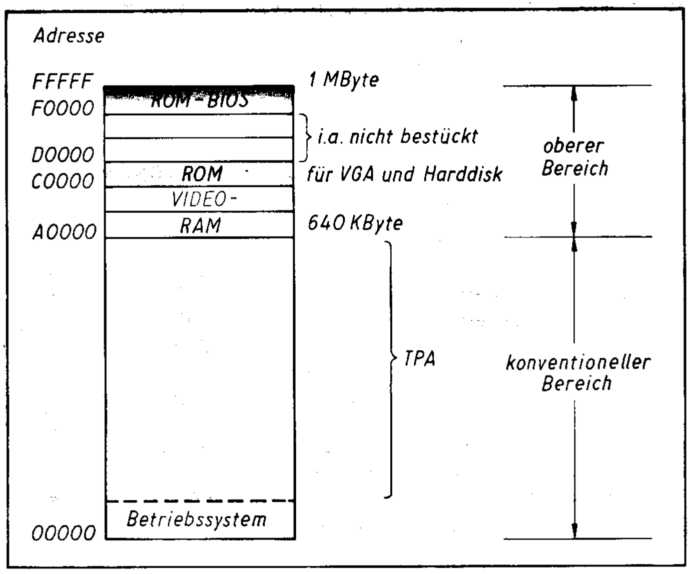


Bild 4 Speicheraufteilung beim XT

Diese Speicherverwaltung „in einem Stück" duldet keine Unterbrechung. Sie macht bei 640 KByte Schluß, weil dort der lineare Bereich des TPA zu Ende ist. Sie ist auch der Grund dafür, daß Speicherblöcke nicht durch Bankswitching in den TPA eingeblendet werden können. Diese „fremden" Blöcke würden natürlich die Ordnung der MCBs durcheinander bringen. So verbleibt für das EMS-Fenster nur der obere Adreßbereich – als Datenspeicher –, denn auf Daten kann ein DOS-Programm ja in jedem Adreßbereich zugreifen.

Mit einem Trick läßt sich die Speicherverwaltung aber doch überlisten, wenn die MCB-Kette passend gepatcht wird. Der TPA wird zunächst durchgehend bis zum Ende (640 KByte) als *belegt* deklariert. Anschließend wird dann den Adapter-Bereichen der MCB für ein Pseudoprogramm zugewiesen. So werden diese Bereiche ausgespart. Ein anschließend geladenes (z. B. speicherresidentes) Programm wird dann automatisch in den oberen Bereich (hinter die Adapter) befördert. Schließlich wird der TPA (durch Änderung des MCBs) wieder als *frei* deklariert, so daß Anwenderprogramme dorthin geladen werden können. Das residente Programm im oberen Bereich bleibt davon unberührt und kann jederzeit aufgerufen werden. Mit Speichermanagern, welche nach diesem Prinzip arbeiten, kann man also auch den RAM des oberen Bereiches für (speicherresidente) Programme nutzen /6/, /7/.

Der RAM des Fensters (64 KByte) wird nicht ausreichen, um mehrere TSR-Programme zu speichern; man kann aber weiteren Expanded Memory dazu verwenden. Bei einem Aufruf werden dann diese Programme aus dem Expanded Memory in das Fenster eingeblendet und dort zum Laufen gebracht. Dazu muß dann nur ein entsprechendes Steuerprogramm resident geladen sein. So wird der TPA spürbar entlastet.

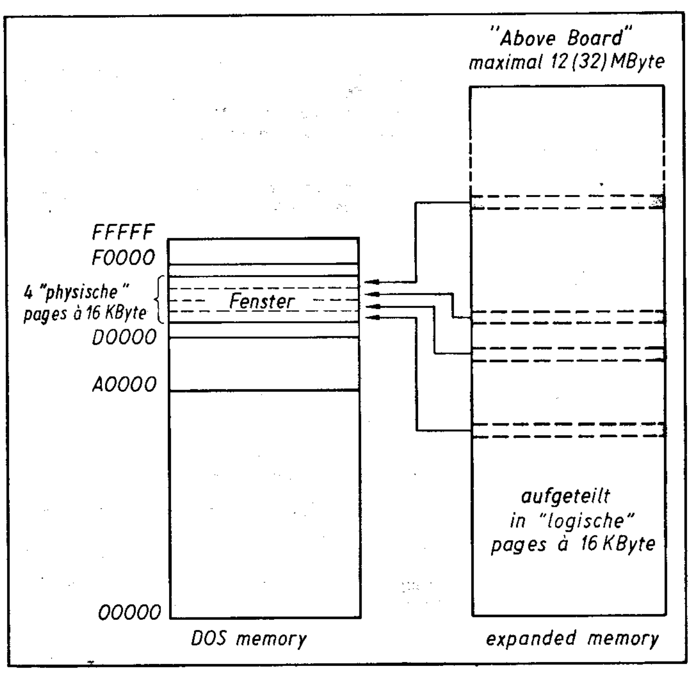
Extended Memory beim AT mit 286er Prozessor

Die Hardware-Entwicklung hat uns weitere Prozessoren beschert, zunächst den 80286 mit einem Adreßbus von 24 Bit, was einen Speicherraum von 16 MByte ergibt. Hier ist also genügend Platz oberhalb der auf dem PC verfügbaren 1 MByte. Man spricht von Extended Memory, welcher nun eine echte lineare Erweiterung des Adreßraumes darstellt (extended – erweitert, hier im Sinne des *verlängerten* Adreßraumes). Für diesen größeren Adreßraum wurde aber auch ein neues Betriebssystem benötigt, da DOS nur 1 MByte verwalten kann. Von Microsoft wurde deshalb das neue Betriebssystem OS/2 (in den Versionen 1.x) entwickelt. Geeignet ist auch das bereits länger existierende Unix.

Allein mit der Entwicklung eines neuen Betriebssystemes ist es aber nicht getan. Das eigentliche Problem ist die Anwendungssoftware für ein Betriebssystem. Bei den Anwendern läuft eine riesige Zahl kommerzieller Programme unter DOS. Diese Programme – sicherlich in Millionen Stückzahlen – müßten bei Einführung eines neuen Betriebssystemes praktisch alle ersetzt werden. Natürlich war es IBM bei der Entwicklung ihres ersten ATs klar, daß die Anwender da nicht mitspielen würden. Also mußte der neue Prozessor in der Lage sein, die vorhandene Software weiterhin zu verarbeiten. Das heißt, der 80286 mußte die Fähigkeit haben, auch mit dem Betriebssystem DOS zu arbeiten. Hardwaremäßig wurde das so gelöst, daß er über zwei verschiedene Betriebsmodi verfügt: den Normal-Modus (Real Mode), in dem ein Speicherraum von 1 MByte in der „klassischen" DOS-Weise adressiert wird, und den geschützten Modus (Protected Mode), in dem ein größerer Speicherraum – also auch Extended Memory – adressiert werden kann (Bild 6). Für den DOS-Betrieb wäre dann allerdings der Extended Memory praktisch tabu.

Ein AT mit dem Prozessor 80286 und (physisch) 1 MByte RAM auf der Hauptplatine hat (logisch) 640 KByte als konventionellen RAM. Es folgt der obere Bereich von 384 KByte (Adapter und ROM), in dem sich kein RAM befindet. Jenseits der (logischen) 1-MByte-Grenze befindet sich der Rest von den (physisch) auf der Hauptplatine vorhandenen 1 MByte (ebenfalls 384 KByte). Das ist der Extended Memory. Kann der stolze Besitzer damit unter DOS etwas anfangen?

Bild 5 Expanded Memory bei XT und 286er AT



Die Softwareentwickler von Microsoft haben den BIOS-Interrupt 15H, der beim ursprünglichen PC für einen Kassettenbetrieb bestimmt war, mit einigen neuen Funktionen wieder zum Leben erweckt. Zunächst kann eine Umschaltung des 80286 vom Real Mode in den Protected Mode durchgeführt werden. Man mußte allerdings feststellen, daß der Prozessor einen Mangel aufweist. Das Zurückschalten kann nämlich nicht über einen Interrupt, sondern nur durch einen Reset ausgeführt werden. Auf die Bedeutung dieses Handicaps kommen wir noch zurück.
Weiter beschert uns der Interrupt 15H noch eine Funktion, mit der man eine gewählte Anzahl Wörter (bestehend aus je 16 Bit) zwischen dem Extended Memory und dem konventionellen Bereich hin und her transferieren kann. Auf diese Weise läßt sich der Extended Memory als RAM-Disk nutzen. Die Installation erfolgt mit einer der DOS-Routinen Vdisk.sys oder Ramdrive.sys mit der Option /E (für Extended Memory).
Um die Nutzung von Extended Memory für Anwenderprogramme unter DOS in größerem Umfang und vor allem einheitlich zu ermöglichen, hat die oben erwähnte LIM-Gruppe XMS geschaffen (Extended Memory Specification). Es handelt sich dabei diesmal nur um einen Software-Treiber (XMM = Extended Memory Manager) – die Hardware ist ja als Extended Memory bereits vorhanden. Dieser Treiber wird wieder mit einer Device-Anweisung eingebunden. Der Aufruf der einzelnen Funktionen erfolgt hier nicht über einen DOS-Interrupt, sondern mittels Far Call /1/, /2/.
Eine Besonderheit des Extended Memory soll noch erwähnt werden. Es handelt sich um den sogenannten HMA (High Memory Area), einen Bereich am Anfang des Extended Memory, welcher prinzipiell durch direkte Adressierung unter DOS erreichbar ist. Es gilt nämlich: Das größte Segment ist FFFF0. Mit einem Offset von 0000F ist dann die 1-MByte-Grenze erreicht. Man kann aber natürlich den Offset vergrößern – bis maximal 0FFFF –, das ergibt dann die Adresse FFFF0 + 0FFFF = 10FFFF. Ab einem Offset von 00010 tritt dabei ein Überlauf nach Bit 20 des Adreßbusses auf. Weil beim 8086/8088 kein Bit 20 existiert, geht dort der Überlauf verloren, und es werden wieder die Adressen ab 0000:0000 angewählt (Wrap Around). Beim 80286 aber kann die höhere Adresse direkt angewählt werden. Voraussetzung ist, daß Extended Memory vorhanden und Bit 20 nicht gesperrt ist. Im Real Mode ist Bit 20 für DOS aus Gründen der Kompatibilität gesperrt. Sperrung und Entsperrung ist softwaremäßig möglich und steht als Funktion unter XMS zur Verfügung. Damit kann dann auch der HMA verwaltet werden.
Trotz aller Bemühungen der LIM-Gruppe, die Nutzung des Extended Memory zu erleichtern, ist dies (im Gegensatz zum Expanded Memory) in kommerziellen Programmen unter DOS kaum möglich. Eines der wenigen Beispiele ist die Version 3.0 von Lotus 1-2-3.

Einige Bezugsquellen

EMS-Einsteck-Karten („Above Boards") Zur Installation von Expanded Memory auf PC/XT und 286er AT	
RAM-Kapazität	ca. 4–16 MByte pro Board. (Für größeren Bedarf können mehrere Boards parallel geschaltet werden.)
Preise Board (unbestückt einschl. Treibersoftware) 1 MByte RAM (80 ns) Bezug	 ca. 250 bis 400 DM. ca. 200 DM. über Fachhandel
EMS-Emulatoren Programme, die – mit Extended Memory auf 80286 oder Harddisk-Speicher auf dem PC/XT – EMS emulieren	
Preise ① Above DISC ② Turbo EMS ③ TC!Power Distributoren	 ca. 235 DM ca. 295 DM ca. 275 DM ①② H+BEDV, Olgastr. 4, W-7992 Tettnang 1 ③ Software Connection, Schulstr. 18, W-8913 Schondorf
Speichermanager für 8086/8088 und 80286 Programme, die es ermöglichen, bei vorhandenem Zusatz-RAM – kein Extended Memory – DOS-Programme „zwischen den Adaptern" zu laden und dort laufen zu lassen.	
Preise ① Qram (von Quarterdeck) ② MOVE'EM (von Qualitas) Distributoren	 ca. 200 DM ca. 285 DM ① TriDis, Hansastr. 15, W-8000 München 21 ② Albrecht Software, Mooswiesenstr. 11 A, W-8000 München 60
Speichermanager für 80386 Programme, die es ermöglichen, – nur auf dem 386er AT – Extended Memory „zwischen den Adaptern" einzublenden und wie für 8086/80286 zu nutzen.	
Preise ① QEMM-386 (von Quarterdeck) ② 386MAX (von Qualitas) Distributoren	 ca. 300 DM ca. 250 bis 350 DM (je nach Version) ① TriDis ② Albrecht Software
DOS-Extender Programme, die es für 286er und 386er gestatten, spezielle DOS-Anwendersoftware über 640 KByte hinaus laufen zu lassen.	
Preise DOS Extender OS/286 (von Eclipse) DOS Extender OS/386 (von Eclipse) DOS Extender 386 (von Phar Lap) Distributor	 ca. 1500 DM ca. 1500 DM ca. 1500 DM Tech Soft, Nibelungenstr. 4, W-1000 Berlin 28

Expanded Memory beim 286er AT

Programme, die Expanded Memory auf einem PC/XT nutzen, sollen natürlich auch auf einem PC/AT laufen. Nun läßt sich aber ein vorhandener Extended Memory bei der 80286-CPU des ATs nicht in den Adreßbereich oberhalb von 640 KByte einblenden. Genau wie beim PC/XT benötigt man ein „Above Board" als Zusatzhardware. Dann kann man außerdem den darauf befindlichen RAM in zwei Gruppen aufteilen und sowohl als Expanded wie als Extended Memory verwenden.
Findige Köpfe haben schließlich doch reine Softwarelösungen ausgeknobelt, um Expanded Memory durch Extended Memory zu emulieren. Das Fenster kann dann aber nicht im oberen, sondern muß im konventionellen Bereich liegen – also unterhalb der 640-KByte-Grenze. Damit gehen 64 KByte für den TPA verloren /8/. Es besteht noch ein weiterer Nachteil gegenüber der Hardwarelösung: Bei der Hardwarelösung wird ein Speicherblock durch Umschalten in das Fenster eingeblendet; dagegen müssen bei der Emulation die Daten vom Extended Memory in das Fenster umgespeichert werden. Das dauert natürlich länger.
Dafür funktioniert aber die Emulation sogar schon, wenn überhaupt kein Extended Memory vorhanden ist. Dann wird dieser nämlich durch Speicherplatz auf der Harddisk ersetzt, so daß Daten zwischen Fenster und Festplatte umgeladen werden. Allerdings braucht das natürlich noch wesentlich mehr Zeit (Zeitfaktor Expanded : Extended : Harddisk etwa 1 : 20 : 2000). Das läuft dann sogar auf einem XT ohne echten Expanded Memory. Sinnvoll ist diese Anwendung allerdings wohl nur, um ein Programm, welches

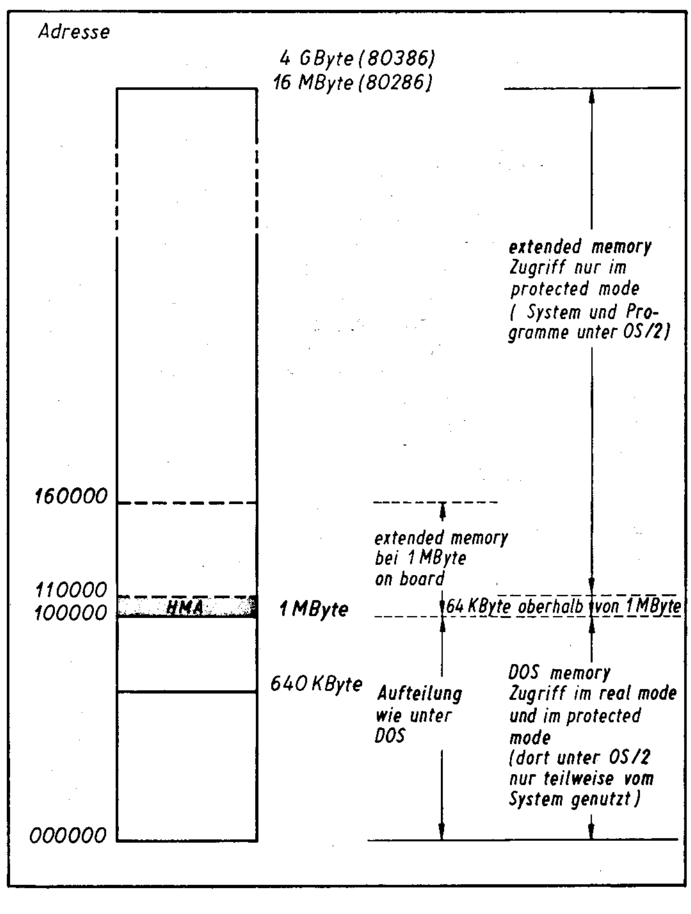


Bild 6 Speicheraufteilung bei 286er und 386er AT

EMS benötigt, überhaupt zum Laufen zu bringen.

Der DOS-Extender beim AT

Der DOS-Anwender wird natürlich fragen, ob man sich nicht doch von den Beschränkungen auf die 640 KByte TPA freimachen kann. Für den eigentlichen Ablauf eines DOS-Programms sind sie ja keineswegs relevant. Doch der ganze Speicherraum von 16 MByte steht nur unter dem Betriebssystem OS/2 im Protected Mode zur Verfügung, und hier läuft kein DOS-Programm. Um es zum Laufen zu bringen, muß auf Real Mode geschaltet werden, und dann sind die Beschränkungen wieder da!

Einen Ausweg aus diesem Dilemma verdanken wir wiederum findigen Programmierern. Er heißt DOS-Extender. Das zugrunde liegende Prinzip ist einfach: Der „reine" Code eines Programms läuft – unabhängig vom Betriebssystem – in jedem (linearen) Speicher, und das Programm kann auf alle Daten in einem beliebig großen Speicher zugreifen, wenn sie adressierbar sind. Die Beschränkungen durch das Betriebssystem treten wesentlich erst dann in Aktion, wenn Systemroutinen aufgerufen werden, zum Beispiel für Ein- und Ausgabe.

Der DOS-Extender läuft unter OS/2 mit Protected Mode. Er läßt auch das DOS-Programm unter OS/2 mit dem ganzen zur Verfügung stehenden Speicherraum laufen, und zwar so lange, bis eine DOS-Systemroutine angesprochen wird. Dann muß er auf die DOS-Umgebung, das heißt in den Real Mode umschalten, bis die Routine beendet ist. Anschließend kann es unter OS/2 weitergehen. Das heißt aber für den 80286: Es muß ständig softwaremäßig zwischen Real und Protected Mode hin- und hergeschaltet werden. Wir haben auf den Prozessormangel bereits hingewiesen, der das eine zwar gestattet, das andere aber (fast) unmöglich macht. Glücklicherweise hat man dann doch eine Softwarelösung gefunden, die es mit vielen Klimmzügen gestattet, den Reset so abzumildern, daß das Programm weiterläuft.

Unter einem DOS-Extender laufen dann auch solche DOS-Programme, die für die normale DOS-Umgebung zu lang sind. Neben einigen kommerziellen Programmen (z. B. CAD-Programme) sind das vor allem Compiler für verschiedene Hochsprachen. Mit diesen kann man dann auch extrem lange Quellprogramme direkt unter einem DOS-Extender compilieren und laufen lassen /9/.

Weitere Verbesserungen mit dem 80386

Der Nachfolger des 80286 ist der 80386. Er verfügt über einen Datenbus und einen Adreßbus von jeweils 32 Bit. Das ergibt einen Speicherraum von 4 Gigabyte. (Ähnlich wie seinerzeit beim 8086/8088 wurde auch eine Version 80386SX entwickelt, die einen Datenbus von nur 16 Bit besitzt, dazu allerdings auch nur einen Adreßbus mit 24 Bit.) Gegenüber seinem Vorgängermodell hat ein mit diesem Prozessor bestückter Rechner verbesserte Möglichkeiten für das Multitasking (gleichzeitiger Ablauf mehrerer Prozesse).

Der PC/XT wurde seinerzeit – nach dem Vorbild der CP/M-Rechner – für Single-User- (Ein-Nutzer-) und Single-Task-Betrieb entwickelt. Dementsprechend sind die meisten DOS-Routinen nicht reentrant, das heißt, während des Ablaufes einer Routine darf kein weiterer Aufruf erfolgen. Das Programm würde „den Faden verlieren" und den Weg zurück durch die Kette der Aufrufe nicht mehr finden.

Beim 286er AT wurde – unter dem Betriebssystem OS/2 – Multitasking (und auch Multiuser-Betrieb im Netz) vorgesehen. Im Protected Mode läuft jedes einzelne Programm (unter der Herrschaft eines zentralen Steuerprogramms) in seiner eigenen *virtuellen* Umgebung – quasi für sich allein, geschützt gegen alle etwaigen Beeinflussungen durch andere Programme. Daher wird der Protected Mode auch Virtual Protected Mode genannt.

Dieses Konzept wurde beim 80386 noch weiter verbessert. Der Prozessor verfügt über drei verschiedene Betriebsmodi. Zunächst – analog wie beim 80286 – über den Real und den Protected Mode. Hinzugekommen ist der Virtual 8086 Mode (V86). In dieser Betriebsart kann eine (nur durch den verfügbaren Speicher begrenzte) Anzahl *virtueller* DOS-Umgebungen eingerichtet werden. In jeder dieser Umgebungen (allerdings innerhalb der „klassischen" 1-MByte-Grenze) kann ein DOS-Programm laufen. Der Benutzer kann entweder jeweils ein Programm anwählen oder alle Programme (unter einem Steuerprogramm) im Multitasking laufen lassen.

Weiterhin verfügt der 386er Prozessor über ein völlig neues Konzept der Speicheradressierung. Die bei den Vorgängermodellen vorhandene feste Kopplung der Adressen an die vorhandenen Speicherblöcke wurde aufgegeben. Der gesamte physische Speicher ist in 4-KByte-Pages (Seiten) aufgeteilt. Jede dieser Pages kann einem beliebigen logischen Adreßbereich von 4 KByte zugeordnet werden. Das ist also „Bank Switching total". In den logischen Adreßraum können die physischen Pages ganz nach Belieben eingeblendet werden. Dann läßt sich auch Extended Memory in den Adreßraum oberhalb von 640 KByte einschalten, und damit ist Expanded Memory ohne Hardwarezusatz im Real Mode möglich (Boardless EMS). Man benötigt nur die entsprechende Software (Speichermanager) /10/.

Als Betriebssystem (für den Protected und V86 Mode) wird künftig ein erweitertes OS/2 (Version 2.0) dienen. Auch Unix ist gut verwendbar. (Der Real Mode läuft – genau wie beim 80286 – unter DOS.) Natürlich hat man den Mangel des 80286 beim 80386 behoben und kann jetzt problemlos softwaremäßig zwischen den einzelnen Betriebsmodi hin- und herschalten. Daher läuft auch der 80386-DOS-Extender schneller und besser. Allerdings kann dabei – im Gegensatz zum V86 Mode – nur eine DOS-Anwendung laufen /9/.

Hat DOS noch eine Zukunft?

Beim Studium der verfügbaren Literatur kann man zu dem Schluß kommen, daß die Tage von DOS gezählt sind. Diese Ansicht ist falsch. Allein die Zahl der immer noch im Einsatz befindlichen Programme garantiert DOS noch ein Leben auf viele Jahre hinaus. Die neue Version 5.0 wird in Kürze auf dem Markt erwartet. Auch heute noch werden neue Programme meist in einer DOS-Ausführung entwickelt. Eine OS/2-Ausführung folgt – wenn überhaupt – nur zögernd.

Das Betriebssystem OS/2 wurde in der Version 1.0 für den 286er AT entwickelt und 1987 von der IBM vorgestellt. Heimisch ist es auf diesem Rechner aber nie geworden. „Serienmäßig" wird praktisch kein Gerät mit OS/2 ausgeliefert. Das ist auch nicht so verwunderlich. Die Version 1.2 für den 80286 benötigt nämlich mindestens 4 MByte RAM. Dazu kommt, daß Anwenderprogramme unter OS/2 immer noch rar (und teuer) sind. Für den 80286 wird sich daran wohl auch nichts mehr ändern.

Anders wird es beim 386er AT werden. Auch hier laufen die Versionen 1.x von OS/2. Die neue Version 2.0 (nicht OS/3, wie vielfach vermutet wurde) ist aber genau auf diesen Prozessor zugeschnitten. Sie unterstützt natürlich auch den DOS-Extender und den Ablauf von DOS-Programmen im V86 Mode.

All das zeigt, daß die DOS-Anwendungen, nicht nur auf den altehrwürdigen PCs, sondern ebenso auf den modernsten Spitzenrechnern, auch in Zukunft den Hauptanteil der eingesetzten Programme ausmachen werden. Dem Nachteil eines zu engen Speicherraumes kann mit den hier gezeigten Möglichkeiten wirksam begegnet werden.

Literatur

/1/ Duncan, R.: MS-DOS Extensions. Microsoft Press 1989.
/2/ Albrecht, H.: EMS 4.0 und XMS 2.0, Alle Funktionen im Überblick. c't, Hannover (1989) 11, S. 379
/3/ Albrecht, H.: Odyssee im Adreßraum. c't, Hannover (1989) 11, S. 336
/4/ Rinke, W.: Über alle Grenzen. toolbox (1990) 3, S. 66
/5/ Gieselmann, K.: Rumtreiben im Speicher. toolbox (1990) 4, S. 66
/6/ Ernst, M.: Treiber verschiebt Treiber. c't, Hannover (1989) 8, S. 130
/7/ Wilde, M.: RAManager, Speichermanager für PCs mit 8088 bis 80286. c't, Hannover (1990) 6, S. 138
/8/ Gieselmann, K.: TC!Power und TurboEMS – Emulatoren im Vergleich. toolbox (1990) 3, S. 10
/9/ Rothe, D.: Bis zu vier Gigabyte, DOS-Extender nutzen den gesamten PC. c't, Hannover (1990) 1, S. 200
/10/ Ellinghaus, R.: Speicherschieber, Memory-Manager 386-to-the-MAX für 80386er. c't, Hannover (1989) 4, S. 74
/11/ PC-Hauptspeichererweiterungen. MP, Berlin 3 (1989) 3, S. 89
/12/ Gieselmann, K.: Speichererweiterungen im PC. toolbox (1990) 3, S. 88

AST-Erweiterungskarten jetzt mit Headroom-Software

AST Research bietet seine EMS-Speichererweiterungskarten jetzt mit der Software Headroom an, die den DOS-Speicher organisiert. Mit Headroom können alle speicherresidenten Programme für Treiber, Anwendersoftware usw. aus dem 640-KByte-DOS-Speicher in den Extended Memory, den Expanded Memory oder auf die Festplatte geladen werden. Die Verbindung von AST-Speicherkarten und Headroom ermöglicht auch, Treiber für die Tastatur oder Netzwerke ins „Highdos" zu laden. So wird der DOS-Speicher frei für umfangreiche Anwendungen. Insgesamt 32 Anwendungen können gespeichert werden (EMS 4.0 ist Standard) und jederzeit per Hotkey aufgerufen werden. Damit kann der Anwender mühelos und schnell in verschiedenen Programmen arbeiten. Er kommt jeweils per Hotkey dort im Programm an, wo er es zuletzt verlassen hat. Er braucht also nicht mühsam weiter zu blättern, bis er an der Stelle seiner letzten Bearbeitung ist. Die Cut-and-Paste-Funktion erlaubt auf einfache Weise, Daten von einem Programm in ein anderes zu transferieren. Auch Anwendungen, die nicht EMS-fähig sind, können durch Headroom aus dem 640-KByte-Speicher ausgelagert werden. *MP*

Computer-Club

Realtime-Clock-Modul für 8-Bit-Rechner

Olaf Matthäi, Königs Wusterhausen

Bisher ist in kaum einem 8-Bit-Rechner eine exakte Echtzeituhr enthalten, die nachfolgende Parameter garantiert:
- Es werden keine Speicherzellen des Grundrechners genutzt.
- keine Fehlzeiten bei Datenübertragung zum Massenspeicher (Diskette oder Kassette)
- geringer finanzieller Aufwand und möglichst geringer Platzbedarf
- Batteriepufferung des RTC (10 µA).

Diese Bedingungen werden durch den von mir entwickelten Modul erfüllt.
Es werden nur 3 ICs verwendet: 74LS00, 8205, RTC 62421. Der RTC 62421 ist in Elektronikgeschäften für etwa 10,– DM erhältlich (die Ausgabe lohnt sich).
Der RTC beinhaltet eine komplette Echtzeituhr (Quarzgenerator, Uhrzeitspeicher mit Sekunde, Minute, Stunde, Tag, Monat, Jahr, Woche und 3 Steuerregistern). Dafür werden 16 I/O-Register benötigt.
Für den Z 1013 wurden die I/O-Adressen 70H-7FH gewählt. Es sind noch folgende Adressen möglich (Z 1013 mit Baugruppenträger): 10H-1FH, 30H-3FH, B0H-BFH, D0H-DFH. Eventuell gibt es bei den gewählten Adressen Probleme mit der Floppykarte (I/O = 7D-7F).
Die Vorteile des Moduls sind:
– keinerlei Abgleicharbeiten
– unkritischer Aufbau
– völlig unabhängig von der Rechnertaktfrequenz.

Schaltung
Die Schaltung ist aus Bild 1 ersichtlich und bereitet sicher keinerlei Probleme. Als Leiterplatte ist ein kleines Stück Raster- oder Streifenleiterplatte ausreichend. Es gibt keine kritischen Punkte in der Schaltung. Die Pufferung und das Weiterverarbeiten der Uhr werden durch einen Akku (4 × 1,2 V) ermöglicht. Da der RTC nur 4-10 µA Strombedarf hat, reicht der Akku zirka 1 Jahr.

Programme in Basic und Maschinencode
Nach dem Aufbau wird der Modul ohne RTC an den Rechnerbus gesteckt und das Basicprogramm gestartet. Nun muß die I/O-LED leuchten. Danach (Betriebsspannung abschalten!) kann der RTC gesteckt werden. Jetzt nochmal das Basicprogramm starten. Nun muß nach der Eingabe der aktuellen Uhrzeit diese ständig aktualisiert werden (auf dem Bildschirm links unten).
Das MC-Programm ist nicht optimiert worden. Es läßt sich sehr leicht in den Monitor des Z 1013 einbauen. Das MC-Programm besteht aus den 2 Teilen INITRTC und RTC.
INITRTC: initialisiert den RTC und fragt die aktuelle Zeit ab. Das Laden der Steuerregister muß in der angegebenen Reihenfolge erfolgen. (Bei Batteriepufferung braucht dieser Teil nicht wieder nach Rechnereinschalten aufgerufen werden.) Dieser Teil kann zum Beispiel unter dem Betriebssystem SCP2RFXXX. unter BOOT aufgerufen werden.
RTC: stellt die aktuelle Uhrzeit in der rechten oberen Ecke des Bildschirms dar (EC17H-EC1FH).
Im Pollingbetrieb wurde dieser Teil unter TSTS aufgerufen. Dadurch wird die aktuelle Zeit immer dann angezeigt, wenn die Tastatur abgefragt wird (INKEY). Das ist bei Bildschirmeingaben und -ausgaben, beim Drukken und anderen Aktivitäten, also 1- bis 50mal in der Sekunde, der Fall. Selbst nach dem Load oder dem Save wird die exakt richtige Zeit angezeigt.
Im Bild 2 ist das Basicprogramm dargestellt, im Bild 3 das INITRTC und im Bild 4 das RTC. Die Programme können beim Autor kostenlos auf Kassette für den Z 1013 überspielt werden (Rückporto).

Tafel 1 I/O-Adressen des RTC-Moduls

Offset	Name	D3	D2	D1	D0	Zählwerte
+0	SECEINER	S8	S4	S2	S1	0–9
+1	SECZEHNER	*	S40	S20	S10	0–5
+2	MINEINER	M8	M4	M2	M1	0–9
+3	MINZEHNER	*	M40	M20	M10	0–5
+4	STDEINER	H8	H4	H2	H1	0–9
+5	STDZEHNER	*	AM/PM	H20	H10	0–2/0–1
+6	DEINER	D8	D4	D2	D1	0–9
+7	DZEHNER	*	*	D20	D10	0–3
+8	MEINER	M8	M4	M2	M1	0–9
+9	MZEHNER	*	*	*	M10	0–1
+A	YEINER	Y8	Y4	Y2	Y1	0–9
+B	YZEHNER	Y80	Y40	Y20	Y10	0–9
+C	WEEKDAY	*	W4	W2	W1	0–6
+D	STEUERREGISTER D					
+E	STEUERREGISTER E					
+F	STEUERREGISTER F					

Die Register können gelesen und beschrieben werden.

```
Z1013+K7659/2,5 Mhz
 # D 700 740
0700 F5 C5 3E 01 D3 7D 3E 00 387
0708 D3 7F E7 02 54 49 4D 45 36A
0710 20 3F FF E7 01 D3 75 E7 475
0718 01 D3 74 E7 01 D3 73 E7 45D
0720 01 D3 72 3E 00 D3 70 D3 39A
0728 71 D3 7D 3E 04 D3 7E 3E 392
0730 00 D3 7F C1 F1 C3 5F F0 516
0738 00 00 00 00 00 00 00 00 000
0740 00 00 00 00 00 00 00 00 000
 #
```

Bild 3 Programm INITRTC

```
Z1013+K7659/2,5 Mhz
 # D 800 850
0800 C5 D5 E5 0E 70 ED 78 0E 470
0808 71 ED 40 0E 72 ED 50 0E 369
0810 73 ED 58 0E 74 ED 60 0E 395
0818 75 ED 68 C6 30 32 1F EC 3FD
0820 78 C6 30 32 1E EC 7A C6 3EA
0828 30 32 1C EC 7B C6 30 32 30D
0830 1B EC 7C C6 30 32 19 EC 3B0
0838 7D C6 30 32 18 EC E1 D1 45B
0840 C1 C9 00 00 00 00 00 00 18A
0848 00 00 00 00 00 00 00 00 000
0850 00 00 00 00 00 20 04 CD 0F1
 #
```

Bild 4 Programm RTC

Literatur
/1/ Kieser, H.; Meder, M.: Mikroprozessortechnik, Berlin: Verlag Technik, 1986
/2/ REALTIME CLOCK MODULE RTC-62421, Application Manul, Seiko Epson Corp.

KONTAKT
Salvador-Allende-Straße 11b,
O-1600 Königs Wusterhausen

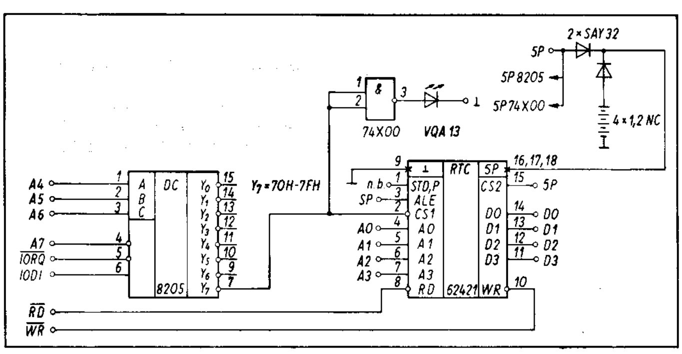


Bild 1 Schaltung des RTC-Moduls mit Akkupufferung

```
1 OUT(125),1:OUT(127),15
2 CLS:PRINT"TIME ?":PRINT:INPUT"H*10";HZ:INPUT"H*1";HE
3 INPUT"M*10";MZ:INPUT"M*1";ME
10 OUT(112),0:OUT(113),0:OUT(114),ME:OUT(115),MZ
20 OUT(116),HE:OUT(117),HZ
40 OUT(125),0
50 OUT(126),4
60 OUT(127),0
65 CLS
70 A=INP(112):B=INP(113)
75 C=INP(114):D=INP(115)
76 E=INP(116):F=INP(117)
80 PRINTAT(30,6);D:PRINTAT(30,8);C:PRINTAT(30,12);B:PRINT   AT(30,14);A
81 PRINTAT(30,0);F:PRINTAT(30,2);E
90 GOTO70
```

Bild 2 Basic-Listing

MS-DOS-Tip

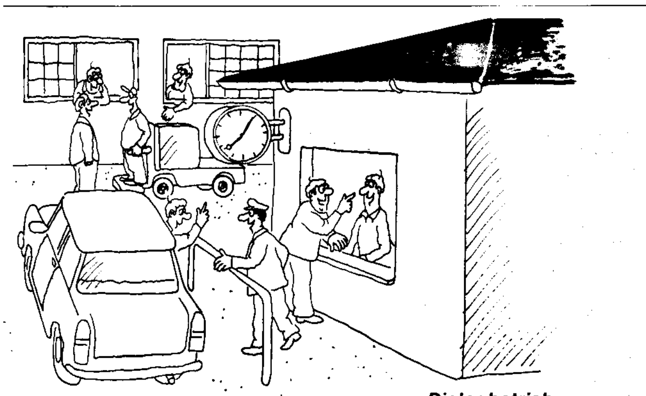
Inhaltsverzeichnisse vergleichen

Oft ist es notwendig, den Inhalt von zwei Verzeichnissen miteinander zu vergleichen, um beispielsweise „Doppelgänger" aufzuspüren, Archiv-Disketten zu überprüfen und auf den aktuellen Stand zu bringen und ähnliches. Die Batch-Datei Cdd.bat löst dieses Problem und kann damit ein sehr nützliches Hilfsmittel sein.
Verglichen wird immer das jeweils aktuelle Verzeichnis mit dem angegebenen Ziel, das ein anderes Verzeichnis oder Laufwerk sein kann. Der Aufruf erfolgt in der bekannten Form der DOS-Befehle:
cdd [Dateispezifikation] [LW:\Pfad\...] [Parameter]
Für die Dateispezifikation gibt es keine weiteren Einschränkungen; sie muß natürlich der DOS-Konvention entsprechen. Laufwerk und Pfad werden ebenfalls in der üblichen Weise angegeben. Vier Parameter stehen zur Verfügung, die aber nicht kombinierbar sind. Im einzelnen bewirken diese folgende Funktionen:
d übereinstimmende Dateien werden im Ziel gelöscht
c fehlende Dateien werden in das Ziel kopiert
+ übereinstimmende Dateien im Ziel werden aufgelistet
– fehlende Dateien im Ziel werden aufgelistet
Bei den Parametern c und d werden die Dateien vor dem Kopieren bzw. Löschen ebenfalls aufgelistet, verbunden mit einer Sicherheitsabfrage, so daß gegebenenfalls mit <Ctrl>-C ein Abbruch bewirkt werden kann.
Ohne Parameterangabe bzw. beim Aufruf mit **cdd ?** wird ein Hilfstext angezeigt. Zum Beispiel werden mit

cdd i*.bat d:\batches c

alle Dateien, die der Spezifikation entsprechen und im Verzeichis d:\batches nicht enthalten sind, aufgelistet und bei Bedarf dorthin kopiert. Eine Einschränkung in bezug auf den Vergleich der Dateien ist insofern zu beachten, da nur nach dem Namen, nicht aber nach dem Inhalt verglichen wird.
Wenn Sie MS-/PC-DOS-Versionen unter 3.30 nutzen, lassen Sie bitte das Zeichen @ vor Echo off weg. Ab Version 3.30 bewirkt es, das auch der Befehl Echo off nicht auf dem Bildschirm erscheint. Leerzeilen innerhalb von Batch-Dateien werden durch **Echo.** erzeugt. Damit Ihnen dieses Programm ständig zur Verfügung steht, sollte es über den in der Autoexec.bat angegebenen Suchpfad erreichbar sein. Falls nicht ausreichend Umgebungsspeicher zur Verfügung steht, können Sie den Kommandoprozessor Command.com mit den Optionen **/p** und **/e:nnn** laden. Dabei gibt nnn den zu reservierenden Speicherbereich an. *Roland Radandt*

```
@echo off
rem (c) R.Radandt -- Klinikum Berlin-Buch, Hs. 505 F
rem Vergleich des Inhaltes von 2 Verzeichnissen
rem Version 2.1
cls
if "" == "%1" goto expl
if ? == %1 goto expl
goto ok
:expl
echo Eingabe-Syntax cdd (compare directory to directory)
echo.
echo cdd [Datei-Spezifikation] [LW:\Pfad\...] [Parameter]
echo Die Wildcards * und ? sind zulässig. Verglichen wird
echo immer das aktuelle Verzeichnis mit dem angegebenen
echo Ziel.
echo.
echo Zulässige Parameter:
echo.
echo    d = übereinstimmende Dateien werden gelöscht
echo    c = nicht übereinstimmende Dateien werden kopiert
echo    + = übereinstimmende Dateien werden aufgelistet
echo    - = nicht übereinstimmende Dateien werden aufgelistet
echo.
echo Beispiele : cdd i*.bat d:\batches c
echo             cdd *.* a: -
goto end
:ok
set MESS=0
if d == %3 goto comp1
if c == %3 goto comp2
if + == %3 goto comp1
if - == %3 goto comp2
goto expl
:comp2
echo Verzeichnis-Vergleich und Auflisten der nicht übereinstimmenden
echo Dateien  %1  in  %2
echo
for %%a in (%1) do if not exist %2\%%a set MESS=1
if 0 == %MESS% goto nofile
for %%a in (%1) do if not exist %2\%%a echo %%a
if c == %3 goto copy
goto end
:comp1
echo Verzeichnis-Vergleich und Auflisten der übereinstimmenden
echo Dateien  %1  in  %2
echo.
for %%a in (%1) do if exist %2\%%a set MESS=1
if 0 == %MESS% goto nofile
for %%a in (%1) do if exist %2\%%a echo %%a
if d == %3 goto del
goto end
:copy
echo.
echo Sollen diese Dateien nach  %2  kopiert werden ?
echo Ctrl-C für Abbruch
echo.
pause
for %%a in (%1) do if not exist %2\%%a copy %%a %2
goto end
:del
echo.
echo Sollen diese Dateien in  %2  gelöscht werden?
echo Ctrl-C für Abbruch
echo.
pause
for %%a in (%1) do if exist %2\%%a attrib -r %2\%%a
for %%a in (%1) do if exist %2\%%a del %2\%%a
echo.
echo Datei(en) wurden gelöscht !
goto end
:nofile
echo Keine Datei(en) gefunden !
:end
set MESS=
```



Dialogbetrieb

Zeichnung: Steger

Schnelle Winkelfunktionen in Turbo-Pascal

Mit der Benutzung eines numerischen Koprozessors lassen sich Gleitkommaoperationen, die bei mathematisch-physikalischen Aufgaben den Hauptteil der Programmlaufzeiten ausmachen, stark beschleunigen. Interessant ist es in diesem Zusammenhang, den vom Turbo-Pascal-Compiler generierten Maschinenkode zu analysieren. Es stellt sich heraus, daß die Gleitkommaeinheit (FPU) nicht optimal programmiert wird. Dafür gibt es hauptsächlich zwei Gründe.

① Der Compiler muß sich aus Kompatibilitätsgründen auf den Befehlssatz des Intel 8087 beschränken. Ist eine andere FPU im System, z. B. Intel 80387 oder IIT 2C87, werden Möglichkeiten des erweiterten Befehlssatzes dieser Prozessoren verschenkt.

② Die acht 80-Bit-Register der FPU nutzt der Turbo-Pascal Compiler nicht optimal aus. So werden Zwischenergebnisse nicht im Stack der FPU gehalten, sondern im CPU-Stack zwischengelagert, was die Routinen verlängert und verlangsamt.

Um einen optimalen Kode zu erzeugen, bieten sich Inline-Statements geradezu an, da die direkte Programmierung der FPU sehr einfach ist. In der Unit FPU (Bild 1) ist das an vier Winkelfunktionen demonstriert.

Der Sinus eines Argumentes im FPU-Stack kann erst von den Prozessoren Intel 80387 und IIT 2C87 direkt mit dem FSIN-Befehl berechnet werden. Die Sinusfunktion in der Unit FPU ist etwa doppel so schnell wie die von Turbo-Pascal erzeugte Variante. Bei der Tangensfunktion, die im Sprachumfang von Turbo-Pascal nicht enthalten ist, ist der Unterschied gegenüber der Möglichkeit y:=sin(x)/cos(x) noch drastischer. Der Umgang mit einer schon etwas komplexeren Formel und den Registern der FPU wird am Beispiel des Arcussinus demonstriert: arcsin(x)= arctan(x/sqrt(1-x*x)). Der Vollständigkeit halber wurde die Arcustangensfunktion aufgeführt, bei der aber nur ein kleiner Zeitgewinn resultiert.

Auf Bereichsprüfungen der Argumente wurde im Interesse der Geschwindigkeit verzichtet. Zu beachten ist, daß der Intel 8087 über den Definitionsbereich hinausgehende Restriktionen bei manchen Funktionen aufweist. Will man die hier vorgestellten Funktionen nicht innerhalb einer Unit nutzen, wo sie durch ein Intersegment-Call angesprungen werden, muß beim Laden des Arguments nur der Offset 4 zum Basepointer BP addiert werden. Die Unit wurde auf einem AT-kompatiblen PC mit den Koprozessoren 80287 und mit IIT 2C87 getestet.

Es lohnt sich fast immer, zeitkritische Gleitkommaoperationen in Schleifen zu optimieren. Der erzielte Zeitgewinn wiegt die Mühe des Programmierens auf.

Hans Heukenkamp

```
{$N+,E-,S-,V-,R-}

unit FPU;

interface

 function sin(x:extended):extended;     {nur mit 80387, 2C87}
 function tan(x:extended):extended;
 function arcsin(x:extended):extended;
 function arctan(x:extended):extended;

implementation

 procedure fld; inline($CD/$37/$6E/$06);
                                         {fld tbyte ptr[bp+06]}
 procedure fstp; inline($CD/$37/$7E/$F6);
                                         {fstp tbyte ptr[bp-0A]}
 procedure fwait; inline($CD/$3D);       {fwait}
 procedure fld1; inline($D9/$E8);        {fld1}

 function sin;
  begin
   fld;
   inline($D9/$FE);                      {fsin, $D9/$FF für fcos}
   fstp;fwait;
  end;

 function tan;
  begin
   fld;
   inline($D9/$F2/                       {fptan}
          $DE/$F9);                      {fdivp st(1),st}
   fstp;fwait;
  end;

 function arcsin;
  begin;
   fld;
   inline($DD/$D1/                       {fst st(1)}
          $DC/$C9);                      {fmul st(1),st}
   fld1;
   inline($D8/$E2/                       {fsub st,st(2)}
          $D9/$FA/                       {fsqrt}
          $D8/$F9);                      {fdivr st,st(1)}
   fld1;
   inline($D9/$F3);                      {fpatan}
   fstp;
   inline($DE/$D9);                      {fcompp st(1)}
   fwait;
  end;

 function arctan;
  begin
   fld;fld1;
   inline($D9/$F3);                      {fpatan}
   fstp;fwait;
  end;

end.
```

Bild 1 Die Unit FPU

Leipziger Herbstmesse 1990

Von Zuversicht und Weinerlichkeit

Mit dem neuen Leitgedanken „Leipzig hat Zukunft – mitten in Europa" versuchten die Veranstalter aus dem Leipziger Messeamt offensichtlich, sich und ihren Gästen in diesen unruhigen Wendezeiten Mut zu machen. Es war aber bestimmt nicht diesem Slogan zu verdanken, daß die mehr als 340 000 Quadratmeter Ausstellungsfläche ausgebucht waren und 200 Interessenten nicht mehr berücksichtigt werden konnten.
Andererseits konnte er auch nicht verhindern, daß das Messegeschäft dann doch sichtbar träge lief. Auch lag die Zahl der Aussteller letztlich nicht bei zunächst genannten 6000, sondern nur bei 4600. Deutlich reduziert war die Ausstellungsfläche von Firmen der DDR und der anderer osteuropäischer Länder. Ein gewisser Ausgleich konnte allerdings dadurch erzielt werden, daß aus den zerfallenden großen DDR-Betrieben und -Kombinaten jeweils zahlreiche kleinere Firmen hervorgingen. Die BRD war mit etwa 45 Prozent der Ausstellungsfläche größtes Ausstellerland. Viele Firmen nutzten diese Messe erstmals zur Präsentation, auch wenn sie nicht in die traditionelle Branchenspezifik der Herbstmesse paßten. So waren etliche Computeranbieter vertreten, die man nur zu einer Frühjahrsmesse erwartet hätte. Bis zu dieser – so es sie in der bekannten Form tatsächlich noch geben wird – wollte man aber wohl nicht warten.
Das „neue Messekonzept" des Messeamtes sieht auch weiterhin jährlich die beiden Mehrbranchenmessen vor, ergänzt um eine breite Palette von Fachmessen. Wer aber als Aussteller oder Besucher Zeit und Mittel für die Leipziger Frühjahrsmesse 1991 (17. bis 23. März), mit den Bereichen Elektrotechnik, Elektronik, Automation, Bürotechnik und Computer, aufbringen soll, wenn fast zeitgleich in Hannover die CeBIT '91 läuft (13.3. bis 20.3.91), ist fraglich. Ähnlich verhält es sich mit der neuen Fachmesse Büro- und Informationstechnik, die vom 25. September bis 1. Oktober 1991 stattfindet und sich damit gegen mehrere im Herbst etablierte westdeutsche Messen der Büro- und DV-Branche behaupten muß. Die von Ministerpräsident de Maizière anläßlich der Messeeröffnung geäußerte Zuversicht, daß „Leipzig seine Position als bewährter internationaler Handelsplatz in der Mitte Europas weiter ausbauen wird und auch im vereinten Deutschland und im künftigen Europa neue wirtschaftliche Bedeutung erhält", klingt doch mehr nach Pfeifen im dunklen Wald als nach fundierter Erkenntnis.
Ebenfalls mit Zukunftsaussichten, und zwar des wirtschaftlichen Entwicklungsprozesses der Ex-DDR, befaßte sich das vom Messeamt und der Redaktion highTech veranstaltete **1. Internationale Technologiesymposium**. In einer Podiumsdiskussion zwischen namhaften Vertretern beider deutscher Staaten aus Politik und Wirtschaft sollten neue Modelle für Technologiekooperationen und die Risiken solcher Zusammenarbeit erörtert werden. Am ergiebigsten, wenn auch übermäßig zeitintensiv, waren die Gedanken des Technologiebeauftragten des Landes Baden-Württemberg, Prof. Johann Löhn. So bot er an, beim unkonventionellen Aufbau von Technologie-Instituten Unterstützung zu leisten, wie es etwa 100 bereits in der BRD gibt.
Die von den DDR-Vertretern Prof. Terpe (ehem. Minister für Forschung und Technologie der DDR), Prof. Mütze (stellv. Generaldirektor Carl Zeiss Jena) und insbesondere Dr. Lorbeer (Geschäftsführer des Forschungszentrums Medizin- und Labortechnik Dresden) geschilderten Probleme für die DDR-Partner bei Kooperationsversuchen glaubten einige bundesdeutsche Zuhörer in der Diskussion leider mit oberlehrerhaften Bemerkungen verniedlichen zu können („Die Wissenschaftler in der DDR müssen sich auch mal selbst kümmern!"). Bezeichnenderweise verstand es am besten der Geschäftsführer der IBM Deutschland GmbH, Herr Esslinger, klarzustellen, daß solche Hinweise nur aus der Unkenntnis der Verhältnisse in der ehemaligen DDR und über das tatsächliche Verhalten der meisten Wissenschaftler resultieren können. Um die genannten Belehrungen nicht zu provozieren, sollte man wohl lernen, berechtigte Ansprüche nicht nur fundiert, sondern auch bestimmt vorzutragen. Das könnte eher den Eindruck von einer „neuen deutschen Weinerlichkeit" in der DDR verhindern.

Von Händlern und Herstellern

Wie bereits erwähnt, bot diese Herbstmesse nicht das gewohnte branchentypische Bild. Die veränderte wirtschaftspolitische Lage nutzten nämlich auch etliche branchenfremde Anbieter, beispielsweise aus dem Computersektor. Deutlich sichtbar war dabei die Dominanz der Händler bzw. Systemhäuser gegenüber den Herstellern. Zu letzteren zählt beispielsweise die Firma **Bull**. Etwas versteckt, auf der Galerie der Halle 11, war sie erstmals mit ihren Produkten auf einer Herbstmesse vertreten. Präsentiert wurden Mikrocomputer der Reihe DPX/2000 (vorgestellt bereits in unserem Messebericht von der LFM 90 in MP 7/90) sowie die von der 100prozentigen Bull-Tochter Zenith entwickelten Desktop-PCs und Laptops. Außerdem trat Bull – auch mit DDR-Kooperationspartnern – als Anbieter von Lösungen vor allem für Handels- und Produktionsbetriebe der mittelständischen Wirtschaft und der öffentlichen Verwaltung auf.

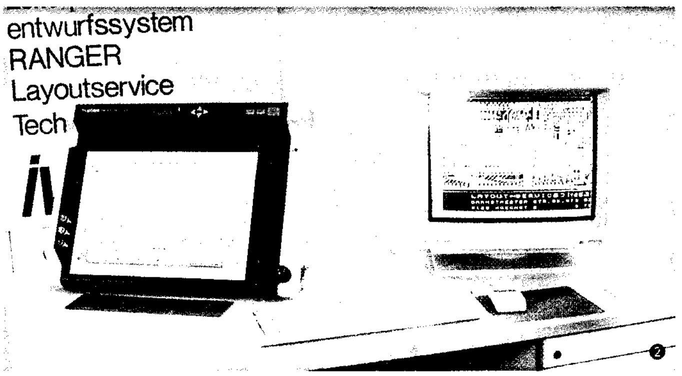
Die **Acer-Cetec Computer GmbH** – erstmals in Leipzig vertreten – präsentierte auf ihrem Stand einen großen Teil ihres Angebotes, vom XT-kompatiblen PC mit 8088-Prozessor bis zum 486er (25 MHz) mit EISA-Bus (Bild 1). Auch nach Abschluß dieser Messe fand man sich in der Auffassung bestätigt, daß das Interesse entgegen landläufiger Meinung auch für XTs noch erheblich ist.

Auf dem Gemeinschaftsstand der USA hatte ein weiterer Produzent Messepremiere: die US-amerikanische Firma **Wang**. Dr. An Wang, 1945 aus China zum Studium der angewandten Physik an die Harvard-Universität gegangen, ist der Gründer dieser Firma. Er erfand 1948 die Magnetimpulssteuerung. Nach diesem Prinzip arbeiteten die Magnetkernspeicher, die bis in die 70er Jahre zu den grundlegenden Komponenten moderner Computer gehörten. Schon ein wenig wie das (fast papierlose) Büro der Zukunft mutete das am Stand von Wang vorgeführte VS Office an. Computer zu vernetzen ist heute schon nichts Aufregendes mehr. Wenn aber meine Visitenkarte über einen Scanner eingelesen und sowohl mit handschriftlichem (über einen Schreibstift, der auch wie eine Maus zum Anklicken benutzt werden kann) als auch mit einem dazugesprochenen Hinweis an eine Mitarbeiterin gesendet wird, finde ich das doch schon bemerkenswert. Zumal alle erforderlichen Operationen über eine angenehme grafische Oberfläche über den bereits erwähnten Schreibstift auszulösen sind. Bedienerfreundlichkeit wird groß geschrieben.
Alles andere, was man von einem Mailingsystem erwartet, ist auch enthalten. So sucht das System automatisch den günstigsten Kommunikationsweg (Telex, Telefax, ...). Neue Mitarbeiter beispielsweise werden in der Filiale, in der sie künftig arbeiten, eingetragen, und die Angaben werden wie alle anderen aktualisierten Daten auch zu einem bei der Installation festgelegten Zeitpunkt allen angeschlossenen Filialen zur Verfügung gestellt. Und wenn Sie meinen, komfortable Textverarbeitung und ein Datenbanksystem gehören auch zu so einem Büroarbeitsplatz, so werden Sie die mit dem VS Office angebotenen Systeme sicher gerne nutzen. Trotzdem können Sie auf Ihrem PC auch weiterhin unter MS-DOS ihre Texte, Datenbanken und Tabellen bearbeiten.

Im Vergleich zu den ausstellenden Hard- oder Softwareherstellern überwogen, wie gesagt, bei dieser Messe eindeutig die Handelsfirmen. Eine mit Tradition in bezug auf Leipziger Messepräsenz ist die West-Berliner Firma **Transcommerz**. Als neue Offerten wurden diesmal das Leasing für Kopierer (353,40 DM), Telefaxgerät (153,90 DM), und den PC AT Tulip compact 3 (306,66 DM; jeweils monatlich, Laufzeit 36 Monate) herausgestellt sowie die Erweiterung der Angebotspalette um Kleinoffsetdruckmaschinen von Ryobi. Sie sollen den Kopierern dort Konkurrenz machen, wo ein hohes Kopiervolumen anfällt.

Die Firma **Grundig** nutzte die Herbstmesse nicht nur zur Konsumgüter-Leistungsschau, sondern war erstmals mit allen Geschäftsbereichen vertreten, also auch mit Industrie- und Büroelektronik sowie PC-Datentechnik. Diese wird über eine 100prozentige Grundig-Tochter, **House of Computers**, bzw. über den Computerfachhandel als Franchisepartner einschließlich Software und Applikationsleistungen angeboten. Herausgestellt wurde in Leipzig insbesondere das Informations- und Dokumentationssystem GBA 2000 auf PC/AT-Basis, das Data-Recorder und WORM-Laufwerke als Massenspeicher nutzt. Auf einer Data-Cassette lassen sich etwa 2000, auf einer der einmal mit Anwenderdaten beschreibbaren WORM-Platte etwa 15 000 DIN A4-Seiten in komprimierter Form elektronisch ablegen.

Eine der zahlreichen westdeutschen Vertriebsfirmen, die bereits Büros in ostdeutschen Städten betreiben, ist die **Keltronic Computervertriebsgesellschaft mbH**. Sie bietet unter anderem 386er und 486er Workstations (auf DOS-Basis) für die Leiterplattenentwicklung mit dem CAE-System Ranger der englischen Firma Seetrax an (Bild 2). Firmen, die Leiterplatten benötigen, können die Entwicklung von Keltronic-Vertragspartnern durchführen lassen oder bei Be-



Produzenten von Hard- und Software der Ex-DDR

neue Firmenbezeichnung	hervorgegangen aus
ARS Aquarius Robotron Systems GmbH	VEB Robotron-Büromaschinenwerk „Ernst Thälmann" Sömmerda
Barock Bürochemie GmbH	VEB Bürochemie Dresden
Bürotechnik Chemnitz GmbH	VEB Robotron-Bürotechnik Karl-Marx-Stadt
Büro- und Datentechnik Vertriebs GmbH	VEB Robotron-Vertrieb Erfurt
Computer-Elektronik Dresden GmbH	VEB Robotron-Elektronik Dresden
Computer-Vertriebs-Union Berlin GmbH (CVU)[1]	VEB Robotron-Vertrieb Berlin
Daimler-Benz InterServices (debis) AG	VEB Leitzentrum für Anwendungsforschung
Data TeleMark (DTM) GmbH	VEB Leitzentrum für Anwendungsforschung
de-pro-ma-electronic GmbH Mühlhausen	VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen
Elektronicon GmbH	VEB Elektronik Gera
Elektro-Apparate-Werke Berlin-Treptow GmbH	VEB Elektro-Apparate-Werke Berlin-Treptow „Friedrich Ebert"
Elektronik Riesa GmbH	VEB Robotron-Elektronik Riesa
Elektroschaltgeräte GmbH	VEB Robotron-Elektroschaltgeräte Auerbach
Elpro AG Berlin	VEB Elektroprojekt und Anlagenbau Berlin
ERMIC GmbH Erfurt	VEB Mikroelektronik „Karl Marx" Erfurt
GKI Gesellschaft für offene Kommunikations- und Informationssysteme mbH	VEB Leitzentrum für Anwendungsforschung
Halbleiterwerk GmbH Frankfurt/O.	VEB Halbleiterwerk Frankfurt/O.
LfA Software und Service GmbH	VEB Leitzentrum für Anwendungsforschung
Meßelektronik Dresden GmbH	VEB Robotron-Meßelektronik „Otto Schön" Dresden
Meßelektronik GmbH Berlin	VEB Meßelektronik Berlin
Pirna Stahlleichtbau GmbH	VEB Robotron-Stahlleichtbau Pirna
Reiss Zeichentechnik GmbH	VEB Robotron Zeichentechnik Bad Liebenwerda
REMA Elektronik GmbH	VEB Robotron-REMA Stollberg
Robotron Anlagenbau GmbH	VEB Robotron-Anlagenbau Leipzig
Robotron Ascota AG Chemnitz[2]	VEB Robotron-Buchungsmaschinenwerk Karl-Marx-Stadt
robotron Automatisierungstechnik GmbH	VEB Robotron-Rationalisierung Weimar
robotron Bürocenter Dresden GmbH[3]	VEB Robotron-Elektronik Dresden
Robotron Büromaschinen AG Sömmerda	VEB Robotron-Büromaschinenwerk „Ernst Thälmann" Sömmerda
Robotron CAMA System GmbH	VEB Robotron-Projekt Dresden
robotron Elektronik GmbH	VEB Robotron-Elektronik Zella-Mehlis
Robotron Erika GmbH	VEB Robotron-Elektronik Dresden
Robotron Export – Import GmbH[4]	VE AHB Robotron Export – Import
Robotron Hoyerswerda GmbH	VEB Robotron-Elektronik Hoyerswerda
Robotron Optima GmbH	VEB Robotron-Optima Büromaschinenwerk Erfurt
Robotron Präzisionstechnik und Elektronik GmbH	VEB Robotron-Goldpfeil-Magnetkopfwerk Hartmannsdorf
Robotron Projekt GmbH	VEB Robotron-Projekt Dresden
Robotron Telecom GmbH	VEB Robotron-Elektronik Radeberg
Software und Systemhaus GmbH	VEB Robotron-Projekt Dresden
Zentrum Mikroelektronik Dresden GmbH	VEB Forschungszentrum Mikroelektronik Dresden

[1] mit Geschäftsbereich IMTRONIC (als GmbH im Aufbau)
[2] mit Unternehmensgruppe FABES (Fabrikationsbetriebe Sachsen)
[3] mit Geschäftsbereich robotron Speichertechnik Dresden und Unternehmensbereich IIS Integrierte Informationssysteme GmbH
[4] mit Geschäftsbereich CVB (Computer Vertrieb Berlin)

darf ein komplettes Entwicklungssystem erwerben. Für das CAE-Programm Ranger wird von Keltronic neben einer Schulung eine 4wöchige Rückgabefrist gewährt.

Noch auf der Leipziger Frühjahrsmesse '90 traten die Betriebe mit dem Label *Robotron* in geschlossener Formation auf. Deshalb fiel die Zuordnung eines Exponates zu seinem Herstellerbetrieb schwer, zumal auch der Handel mit Produkten des ehemaligen Kombinates Robotron wenigen Monopolbetrieben vorbehalten war. Entgegen dem von Ex-Generaldirektor Wokurka zur Frühjahrsmesse vorgestellten Plan, die Robotron-Betriebe unter dem Dach einer Holding-Gesellschaft weiter zu vereinen, zerbröckelte das Kombinat mit seinen damals rund 68 000 Mitarbeitern und 21 Betrieben. Zur Zeit existieren 2 Aktiengesellschaften und 27 GmbHs, die die Robotron-Nachfolge angetreten haben. Einen Überblick über ehemalige DDR-Betriebe (soweit es während unseres Messebesuches möglich war), die einen wesentlichen Anteil an der Produktion von Hard- und Software hatten, finden Sie in der nebenstehenden Tafel.

Während einige Robotron-Firmen weiter auf ihr angestammtes Markenzeichen setzen, glauben andere, daß die Verwendung des alten Labels wegen der früheren Geschäftspraktiken von Robotron eher rufschädigend ist. Entscheidend dürfte aber sein, wie sich jede dieser Firmen ihren neuen Ruf erarbeitet und mit welchen Methoden sie sich auf die Marktbedingungen einstellt. Das wird auf sehr unterschiedliche Weise versucht: Einige widmen sich ganz dem Vertrieb von Waren westlicher Hersteller in der Ex-DDR und dem Noch-RGW, andere stellen die eigene Produktion (neben dem Vertrieb für ihre Kooperationspartner) in den Vordergrund. Beispielgebend scheint hier die **Robotron Büromaschinenwerk AG**, Sömmerda, zu sein. Zur Zeit werden dort zugekaufte Rechnerbaugruppen in die vorhandenen EC 1834/EC 1835-Gehäuse montiert und als neue Produktlinie unter dem neuen (alten) Markenzeichen *soemtron* angeboten. Diese Produktlinie umfaßt PCs mit den Prozessoren 80286 (12 MHz), 80386SX (16 MHz) und 80386 (25 und 33 MHz), die unter anderem eine 40-MByte-Festplatte und einen 14-Zoll-VGA-Monitor enthalten. Die Geschäftsführung hofft, das zugekaufte Know-how schrittweise durch Eigenentwicklungen ablösen zu können und auf diese Weise an europäische Spitzenfirmen heranzukommen. Parallel dazu wird der Vertrieb von PCs aus der Montageproduktion der Tochterfirmen **Aquarius Robotron Systems GmbH** (53 % Büromaschinenwerk, 47 % Aquarius Systems) übernommen. Hierzu gehören Desktop-PCs der Klassen 286 bis 486 sowie ein 286er Laptop. Ähnlich wie bei den PCs wird bei Kassensystemen von NCR und Druckern von Mannesmann/Tally verfahren. Ein Lichtblick sollte jedoch auch die Eigenentwicklung des 24-Nadel-Farbdruckers K 6330 sein (Bild 3). Er kann im Grafikmodus mit der sehr guten Auflösung von 360 Punkten pro Zoll und im Textmodus mit einer Geschwindigkeit von 240 Zeichen pro Sekunde (bei 10 Zeichen pro Inch) drucken. Sein Speicher hat eine Größe von 32 KByte, und er kann zusätzlich Schriftarten von steckbaren Speichermodulen laden. Seine Produktion soll im Februar anlaufen.

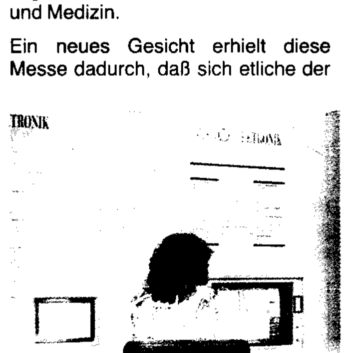
Die zweite im Bunde der Aktiengesellschaften ist die **Robotron Ascota AG Chemnitz**. Sie hat neben Rechnern von C&E (Taiwan), Monitoren von datagraph und 3¹/₃- und 5¹/₄-Zoll-HD-Floppylaufwerken von TEAC unter anderem die 5¹/₄-Zoll-Floppylaufwerke K 5601 aus eigener Produktion im Lieferprogramm.

Bei der **Robotron Elektronik GmbH**, Zella-Mehlis, können beispielsweise Reißwölfe und die einstmals so raren 5¹/₄-Zoll-Festplatten K 5504 mit Kapazitäten zwischen 21 und 68 MByte bezogen werden.

Ausschließlich dem Vertrieb scheint sich die **Computer-Elektronik Dresden GmbH** zu widmen; der Schwerpunkt wird hier auf Siemens-Produkte gelegt. Aber auch der ehemalige Außenhandelsbetrieb Export – Import beweist seine Überlebensfähigkeit: In der Halle 12.12 stellten 12 Firmen, für die die **Robotron Export – Import GmbH** den Vertrieb – wiederum im Osteuropa-Raum – übernommen hat, aus. Ihr Produktspektrum reicht von PCs und ergonomischen Bildschirmen von egs über CAD-/CAM-Systeme von Macrotron bis hin zu Papier- und Bürobedarf von Atlanta und Regis.

Von den Bauelementeherstellern der ehemaligen DDR war mit einem eigenen Stand nur die **Zentrum Mikroelektronik Dresden GmbH** vertreten (Bild 4). Entgegen den Beteuerungen führender Zeiss-Manager, wie Klaus-Dieter Gattner und Wolfgang Nordwig, zur Leipziger Frühjahrsmesse, die Megabit-Schmiede weiter für Entwicklung und Produktion von anwendungsspezifischen Schaltkreisen (ASICs) zu nutzen, wurde das ZMD weder in die Carl Zeiss GmbH mit ihren 11 Werken noch in die ihr unterstehende Holding mit weiteren 6 GmbHs integriert. Vielmehr gehört das ZMD zu jenen 6 Firmen, die aus dem Zeiss-Verband völlig ausgeschieden sind (darunter auch Pentacon Dresden und Elektromat Dresden). Das ZMD mit seinen rund 3000 Mitarbeitern setzt weiterhin auf seine Erfahrungen bei der Schaltkreisentwicklung. Bei den dynamischen RAMs wird zwar die uneffektive Produktion von 1-MBit-Chips eingestellt, aber die Herstellung von 256-KBit-Chips soll noch so lange fortgesetzt werden, wie Absatzmöglichkeiten vorhanden sind. Außerdem werden schnelle statische RAMs mit Kapazitäten von 1 bis 64 KBit (z. B. U 6548) produziert. Das bewährte Design von Standardzellen U 1600 (analog/digital inkl. RAM, ROM und PLA) und Gate-Arrays U 5300 (bis zu 8160 Gatteräquivalente, RAM, ROM) wird weiterhin angeboten. Im Programm von ZMD befinden sich außerdem Cr-Masken nach Kundenwunsch, Entwurfs- und Simulationssoftware, Meßplätze, Sensoren sowie Diagnosegeräte für Mikroelektronik, Umwelt und Medizin.

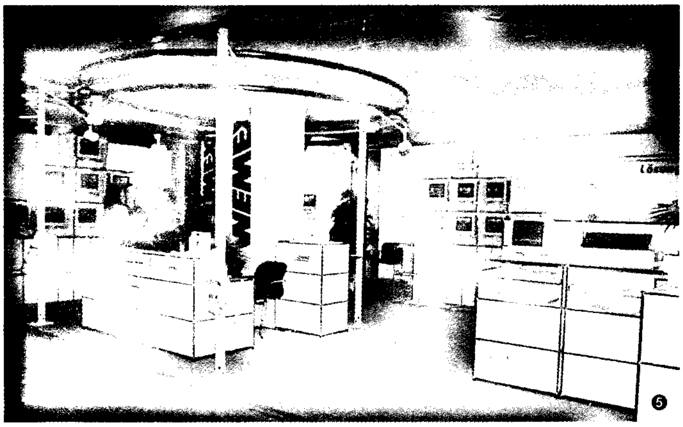
Ein neues Gesicht erhielt diese Messe dadurch, daß sich etliche der



zahlreichen DDR-Existenzgründer einen Messestand leisteten (und hoffentlich auch leisten konnten). Auf die **Wemex Computer Handelsgesellschaft mbH** dürfte letzteres mit hoher Wahrscheinlichkeit zutreffen (Bild 5). Sie konnte allein von März bis Mai dieses Jahres ein Umsatzvolumen von 7 Millionen Mark verbuchen, und bedeutende Exportaufträge in die UdSSR sind bereits gesichert. Das Unternehmen war in den Novembertagen '89 von Mitarbeitern des Datenverarbeitungszentrums des WMW-Außenhandels gegründet und im Februar ins Handelsregister eingetragen worden; der größte Teil der jetzt etwa 40köpfigen Mannschaft sind Kollegen aus dem gleichen Bereich. Obwohl Wemex mit Produkten namhafter Hersteller – wie Compaq, Epson, Triumph Adler – handelt, versteht man sich jedoch nicht nur als Lieferant. Das jetzt schon vorhandene Know-how ist die Grundlage für Angebote von kompletten Lösungen entsprechend den Kundenproblemen. Eingeschlossen sind also Softwareentwicklung, Beratung, Finanzierung, Schulung, Installation, Instandhaltung usw. Auf dem Messestand wurden beispielsweise Lösungen für eine komplette Büroausstattung und für die Vernetzung gezeigt. Als Hochleistungs-Server kann ein Compaq-Systempro dienen, der in einer eindrucksvollen Demonstration zur CeBIT'90 als 386er 250 PCs bediente; jetzt wird auch die 486er Version angeboten.

Ein weit gefächertes Leistungsangebot ist auch bei der **Inframat GmbH** zu finden. Das ebenfalls im DDR-Umwandlungsboom entstandene etwa 200köpfige Unternehmen bietet umfangreiche Leistungen für den Einsatz von PCs, Netzen sowie betriebswirtschaftlicher und Branchensoftware, für die flexible Automatisierung und für die komplette Rationalisierung von Betriebsstätten an. Die Inframat GmbH mit Sitz in Leipzig, Dresden und Berlin ist aus dem VEB Ingenieurbetrieb für Rationalisierung und aus Teilen des ehemaligen VEB Projektas hervorgegangen. Sie arbeitet eng mit dem (DDR-)Jungunternehmer Hans-Holger Kirchhoff und seiner **KCS Kirchhoff Computer-Service GmbH** zusammen. Beide Unternehmen betreuen arbeitsteilig ihre Kunden, wobei der Schwerpunkt bei KCS auf der Hardware liegt. Das Bild 6 zeigt beispielsweise einen DTP-Arbeitsplatz auf der Basis von Siemens-Technik mit dem Scanner HighScan 40, dem 386er Rechner PCD-3T und dem Laserdrucker PT 10. Als Software wurde PageMaker 3.0 unter Windows 3.0 angeboten.

Am Stand der 1988 gegründeten **Condatec AG** war eines der im zurückliegenden Jahr geborenen deutsch-deutschen Joint-venture-Kinder, die Gesellschaft für Informations- und Kommunikationssysteme – die **GIK GmbH** –, zu finden. Die Anteile dieser reinen Vertriebsfirma werden zu gleichen Teilen von der Condatec und vom Datenverarbeitungszentrum Potsdam gehalten. Condatec selbst ist ein Systemhaus für integrierende Lösungen, so zum Beispiel für die Verbindung von DECnet und SNA. Dabei kann die Condatec auf mehr als 10jährige Erfahrungen mehrerer Tochterfirmen in der Hard- und Softwareentwicklung zurückgreifen.



Ihre Devise ist, herstellerunabhängig für jeden Kunden die richtige Lösung zu suchen. Das schließt jedoch Kooperationen mit NCR, Motorola und Siemens nicht aus.

Die junge Leipziger Hard- und Software-Vertriebsfirma **Ohetronic** stellte im Stentzlers Hof neben anderen Produkten wie dem Scanner-Kassensystem Schneider Data Cash, O-Tron-PCs und anderem auch den HP LaserJet IIID der Firma Hewlett Packard als eine der ersten öffentlich vor. Dieser Laserdrucker kann nicht nur wie sein Vorgänger IID auch zweiseitige Ausdrucke herstellen, sondern beherrscht auch die neuentwickelte Resolution Enhanced Technology (RET), die es ermöglicht, bei gleicher Auflösung eine deutlich höhere Druckqualität zu erreichen. Durch einen speziellen anwendungsspezifischen Schaltkreis werden die 49 einen Punkt umgebenden Punkte untersucht und berechnet. Eine schräge Linie beispielsweise wird dann nicht mehr durch Linien gleichgroßer Punkte, sondern durch Doppellinien aus Punkten verschiedener Größe dargestellt. Dies verbessert auch die Qualität des Ausdruckes von Buchstaben mit Serifen (ABCabc) und vermeidet Tonerklumpen auf dem Papier. Einige weitere Leistungsmerkmale des neuen HP-Lasers sind unter anderem eine Druckgeschwindigkeit von bis zu acht Seiten pro Minute, 14 feste und acht skalierbare Schriften sowie die Drucker- bzw. Plottersprachen HP PCL 5 und HP-GL/2. Als Option ist auch Postscript möglich. Der nutzbare Speicher beträgt 720 KByte und kann auf 5 MByte aufgerüstet werden. Der HP LaserJet IIID kann sowohl über eine RS-232-C- als auch über eine Centronics-Schnittstelle angeschlossen werden, optional sind Zusätze für Apple-Talk, HP-LAN und Novell-Netzwerke. Er kostet etwa 9000,– DM.

Von Profisatz und DTP

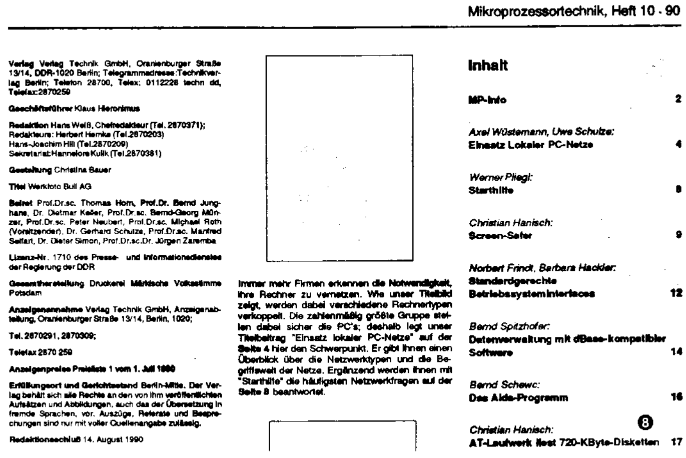
In diesem Abschnitt wäre nun doch noch von einem Bereich zu berichten, der auf der Herbstmesse Tradition hat und durch den Einsatz der Computertechnik auch für uns von Interesse ist: die sogenannte Druckvorstufe. Zu verstehen ist darunter die satztechnische Aufbereitung von Manuskripten und Bildern für den Druck, die früher fast ausschließlich in den Druckereien erfolgte. Mit der Verbreitung des Desktop Publishing (DTP) bieten sich seit einigen Jahren auch Verlagen, Grafikstudios und Herstellerfirmen für bestimmte Produkte kostengünstige Alternativen; zumal die DTP- bzw. Gafiksysteme immer leistungsfähiger werden.

Von den bekannten Herstellern von Satzsystemen für den professionellen Bereich waren auch in diesem Jahr praktisch alle vertreten: **Monotype, Berthold, Linotype, Agfa/Compugraphic** und **Hell/Siemens**. Letztere Firma zeigte erstmals auch ein Satzsystem für Redaktionen: Das integrierte System Cicero ist für den Einsatz auf PCs und in Netzwerken gedacht. Die Texterfassung wurde der redaktionellen Arbeit sehr genau angepaßt: rechts der Text und links die Auszeichnungen. Nachdem die Formate der einzelnen Texte festgelegt sind, kann mit Cicero eine komplette Seite schnell und übersichtlich zusammengebaut oder auch mal umgebaut werden. Eine Verbesserung der redaktionellen Arbeit verspricht auch die Erweiterung CIC-AGENTUR: Aktuelle Meldungen der Nachrichtenagenturen werden über CIC-AGENTUR empfangen und sofort zur Recherche netzweit zur Verfügung gestellt. Die Recherche, die sich auch als Auftrag hinterlegen läßt und permanent wirkt, kann gleichzeitg in mehreren Agenturen mit mehreren Suchkriterien – auch als Volltextsuche – erfolgen.

Ein Hersteller professioneller Satzsysteme ist auch **AM International**. Aber neben dem Angebot der leistungsfähigen Satzanlage EPICS mit 1 Zentraleinheit und bis zu 8 Erfassungsplätzen sowie maximal 9 Ausgabegeräten (einschließlich Postscript) wurde für DTP die Kopplung der Fotosatzbelichter Varityper Serie 4000 mit der Mac-Welt demonstriert; auf dem Mac lief bereits die neue Version 4.0 von PageMaker. Die Kosten für das modulare Belichtersystem belaufen sich auf Werte zwischen 59 000,– Mark für eine Auflösung von 1200 dpi (Punkte pro Zoll) und 145 000,– Mark für 3600 dpi; für ein komplettes DTP-System kommen noch die Kosten für Rechner, Laserdrucker und eventuell Scanner hinzu.

Vielen unserer Leser dürfte – nach der Vorstellung in MP 4/90 – noch das Systemhaus **Mikado** in Erinnerung sein. Nach den veränderten Bedingungen nutzte auch diese Firma jetzt die Leipziger Messe für eine Präsentation ihrer Leistungen – und zwar gemeinsam mit ihren Kooperationspartnern (Bild 7). Der neueste davon war **COBRA**, die Gesellschaft für Computer- und Branchenlösungen mbH, die von fünf Mitarbeitern eines ehemaligen Außenhandelsbetriebes der DDR gegründet wurde. Gemeinsam will man in den Bereichen Bürokommunikation, DTP und betriebswirt-

schaftlichen Branchenlösungen wirksam werden.
Ein weiterer DDR-Partner ist das **Ausbildungszentrum Polygraph-Projekt Chemnitz** (AZP), eine gemeinsame Einrichtung der TU Chemnitz und der Polygraph-Projekt GmbH Leipzig. In- und ausländische Fachkräfte, technische Führungskräfte aus Druckereien und Studenten der Drucktechnik können hier professionelles Publishing erlernen. Ebenso wird aber auch kleinen und mittelständischen Unternehmen die Ausbildung für wettbewerbsfähige Arbeiten mit DTP geboten. Zur Unterstützung dieser Ausbildung hatten Mikado und **Macrotron** übrigens während der Messe in einer feierlichen Übergabe dem AZP zwei DTP-Systeme zur Herstellung von Druckvorlagen und -erzeugnissen zur Verfügung gestellt.
Markstein aus Pfungstadt bot für die Textaufbereitung vom DTP zum Fotosatz ein komplettes Programmsystem an. Mit dem Satzprogramm MIKROSET 300 können bereits auf einem PC/AT Arbeiten ausgeführt werden, die auf der Linotronic-300-Serie möglich sind. Das heißt, es kann satzidentisch erfaßt und umbrochen werden. Eingeschlossen sind eine satzorientierte Texterfassung mit Rechtschreibprüfung und die Datenkonvertierung formatierter Texte von den bekannten Textprogrammen.
Auch das Beratungs- und Systemhaus für professionelles Publizieren **KPS Martin Keller** aus Münster hatte für diesen Bereich etwas zu bieten: Mit Dakon.kann jeder IBM-kompatible PC in einen Fotosatz-Erfassungsplatz für die CRTronic oder das CRTerminal der Serie 300 verwandelt werden. Texte können am PC mit den üblichen Textprogrammen erfaßt und danach mit Dakon übertragen und konvertiert werden – einschließlich aller Satzbefehle, IWT und Makros. Benötigt werden lediglich das Programm und ein Verbindungskabel. Aber auch die Verbindung von DTP und CRTronic ist nun möglich: Fertige Seiten, die mit Ventura Publisher 2.0 und Profi-Erweiterung gestaltet wurden, können mit Publikon umgewandelt und mit Dakon zur CRTronic und damit zur Belichtung geschickt werden. Es ist allerdings gegenwärtig noch nicht möglich, Formeln aus dem Ventura auf der CRTronic auszugeben – diese müßten dort in herkömmlicher Weise gesetzt werden.
KPS arbeitet zur Zeit an der Übertragung des in unseren Landen noch relativ unbekannten **Publishing-Systems 3B2** der britischen Advent Desktop Publishing Ltd. Es läuft nicht nur auf Sun- und Apollo-Workstations, sondern auch auf kompatiblen PCs ab 386-Prozessor. Versprochen werden für 3B2 Leistungsmerkmale, die deutlich über denen der bekannten DTP-Systeme liegen sollen. Das System soll in deutscher Version noch in diesem Jahr zu haben sein.
Letztendlich sei noch von einem **Praxistest** in Sachen DTP berichtet: Anhand von Manuskriptseiten und Layouts der MP 10/90 ließen wir uns einmal die Leistungen des Ventura Publisher 2.0, Profi-Version, vorführen. Auf der Basis eines HP Vectra 486 konnten durchaus ansprechende Ergebnisse erzielt werden. Einbezogen wurde in den Test auch die Kopplung mit einem Linotype-System der Serie 300 und die Ausgabe auf dem Belichter. Probleme gab es bei dieser Zusammenarbeit lediglich beim Formelsatz (siehe oben). Das Ergebnis einer „normalen" Probeseite, versuchsweise auch auf dem Belichter ausgegeben, sehen Sie in Bild 8 als Ausdruck auf dem Laserdrucker. *MP*


Mikroprozessortechnik, Heft 10 · 90

Verlag Verlag Technik GmbH, Oranienburger Straße 13/14, DDR-1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon 28700, Telex: 0112228 techn dd, Telefax: 2870259

Geschäftsführer Klaus Hieronimus

Redaktion Hans Weiß, Chefredakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2870203) Hans-Joachim Hill (Tel. 2870209) Sekretariat: Hannelore Kulik (Tel. 2870381)

Gestaltung Christina Bauer

Titel Werkfoto Bull AG

Beirat Prof. Dr. sc. Thomas Horn, Prof. Dr. Bernd Junghans, Dr. Dietmar Keller, Prof. Dr. sc. Bernd-Georg Münzer, Prof. Dr. sc. Peter Neubert, Prof. Dr. sc. Michael Roth (Vorsitzender), Dr. Gerhard Schulze, Prof. Dr. sc. Manfred Seifart, Dr. Dieter Simon, Prof. Dr. sc. Dr. Jürgen Zaremba

Lizenz-Nr. 1710 des Presse- und Informationsdienstes der Regierung der DDR

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

Anzeigenannahme Verlag Technik GmbH, Anzeigenabteilung, Oranienburger Straße 13/14, Berlin, 1020;

Tel. 2870291, 2870309;

Telefax 2870 259

Anzeigenpreise Preisliste 1 vom 1. Juli 1990

Erfüllungsort und Gerichtsstand Berlin-Mitte. Der Verlag behält sich alle Rechte an den von ihm veröffentlichten Aufsätzen und Abbildungen, auch das der Übersetzung in fremde Sprachen, vor. Auszüge, Referate und Besprechungen sind nur mit voller Quellenangabe zulässig.

Redaktionsschluß 14. August 1990

Immer mehr Firmen erkennen die Notwendigkeit, ihre Rechner zu vernetzen. Wie unser Titelbild zeigt, werden dabei verschiedene Rechnertypen verkoppelt. Die zahlenmäßig größte Gruppe stellen dabei sicher die PC's; deshalb legt unser Titelbeitrag "Einsatz lokaler PC-Netze" auf der Seite 4 hier den Schwerpunkt. Er gibt Ihnen einen Überblick über die Netzwerktypen und die Begriffswelt der Netze. Ergänzend werden Ihnen mit "Starthilfe" die häufigsten Netzwerkfragen auf der Seite 8 beantwortet.

Inhalt

MP-Info 2

Axel Wüstemann, Uwe Schulze: **Einsatz Lokaler PC-Netze** 4

Werner Pliegl: **Starthilfe** 8

Christian Hanisch: **Screen-Safer** 9

Norbert Frindt, Barbara Hackler: **Standardgerechte Betriebssysteminterfaces** 12

Bernd Spitzhofer: **Datenverwaltung mit dBase-kompatibler Software** 14

Bernd Schewc: **Das Aida-Programm** 16

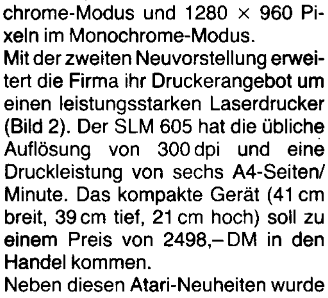
Christian Hanisch: **AT-Laufwerk liest 720-KByte-Disketten** 17

❽

Atari-Messe 1990

❸

Atari-Messe – das bedeutet natürlich auch eine Verpflichtung für den Hersteller, seiner Nutzergemeinde Neues zu präsentieren. Vor allen Dingen ist sie jedoch ein Forum, um zu zeigen, was man mit Atari-Computern machen kann, und wie man es erreicht. Vom 24. bis 26. August dieses Jahres präsentierten in Düsseldorf also 209 Aussteller (etwa 30 Prozent mehr als im Vorjahr) all das an Hard- und Software, was es rund um die Ataris gibt – auch in Konkurrenz zu Atari selbst. Den Worten von Geschäftsführer Alwin Stumpf zufolge gehört es ja zum Selbstverständnis der Firma, die kreativen Potenzen der Anwender zu fördern und – gegebenenfalls in Kooperation – für Weiterentwicklungen zu nutzen. Hier wäre übrigens ein Betätigungsfeld für Entwickler der Ex-DDR. Die Vielzahl von Ausstellern ist demzufolge gern gesehen, da sie den Wettbewerb beleben und zu höheren Leistungen anspornen – auch wenn es bei der Werbung für Alternativen manchmal nicht zimperlich zugeht („Was Atari bisher nicht geschafft hat, bietet jetzt...").
Zunächst jedoch zu den Neuheiten des Herstellers. Hier ist vor allem der TT zu nennen (Bild 1). Aufmerksame Leser werden sich vielleicht daran erinnern, daß wir ihn als Nachfolger des ST anläßlich der Atari-Messe '89 bereits in MP 12/89 vorstellten. Nun soll er aber wirklich in den Handel kommen. Und zwar nicht mit der damaligen 16-MHz-Version des Motorola 68030, sondern mit 32 MHz! Diese Leistungssteigerung bedingte eine vollständige Überarbeitung des TT-Konzeptes. Die weiteren Daten in Kürze: auf dem Prozessorchip integrierte MMU (256 Bit als Cache), Gleitkommaprozessor 686881/68882 optional, eine Harddisk mit mindestens 40 MByte, ein Floppylaufwerk, serienmäßig 2,6 oder 8 MByte Hauptspeicher, mit 4-MBit-RAMs auf 26 Byte erweiterbar, eine parallele und zwei serielle Schnittstellen, MIDI-Anschlüsse, ACSI-DMA-Kanal, SCSI-Schnittstelle, wahlweise 2 SDLC-Schnittstellen oder einen SDLC- und einen LAN-Anschluß; Soundsystem mit 8-Bit-PCM-Stereo, interner VME-bus-Steckplatz. Die Grafikmodi erlauben Auflösungen des Monitors von 320 × 480 Pixeln (256 Farben aus 4096), 640 × 480 Pixeln (16 Farben aus 4096), 640 × 400 Pixeln im Duochrome-Modus und 1280 × 960 Pixeln im Monochrome-Modus.
Mit der zweiten Neuvorstellung erweitert die Firma ihr Druckerangebot um einen leistungsstarken Laserdrucker (Bild 2). Der SLM 605 hat die übliche Auflösung von 300 dpi und eine Druckleistung von sechs A4-Seiten/Minute. Das kompakte Gerät (41 cm breit, 39 cm tief, 21 cm hoch) soll zu einem Preis von 2498,– DM in den Handel kommen.
Neben diesen Atari-Neuheiten wurde


❷

eine Vielzahl von Produkten an den Ständen anderer Firmen angeboten. Das Spektrum der Angebote reichte dabei unter anderem vom Design über Bildverarbeitung, Branchenlösungen, Datenbanken, MIDI, Netzwerke bis zur Zeiterfassung.
Von Vortex beispielsweise wurde ATonce, der AT-Emulator für den Atari ST angeboten;
von X/software das System X/ST/window, ein Fileserver, der auf dem Atari ST basiert und mit einer Implementierung der X-Window-Version 11.4 arbeitet;
von Trade iT – wie von vielen anderen – eine Speichererweiterung für den ST und Mega ST (mit 1-MBit-CMOS-DRAMS bis zu 4 MByte);
von Eickmann Computer als Ablösung des TURBO-16-Beschleunigers der MACH 16 sowie eine Erweiterung, die den ST auf etwa 360 Prozent beschleunigen soll;
von MAKRO C.D.E. für den ST eine Beschleunigerkarte mit 68030-Prozessor für 16 bis 50 MHz, mit der sogar bis zum 20fachen der ST-Geschwindigkeit versprochen wird.
Ein Schwerpunkt war die Musikszene, für die Atari-Computer mit der serienmäßigen MIDI-Schnittstelle ja seit langem prädestiniert sind, ein anderer das Desktop Publishing (DTP). Hierfür war extra ein DTP-Sonderstand eingerichtet worden (Bild 3).
Einen der Hauptanziehungspunkte bildete zweifellos der Stand von DMC, wo mit Calamus S und Calamus SL die Weiterentwicklungen des bekannten Satz- und Layoutsystems gezeigt wurden. Während „S" die Nachfolgeversion des Release 1.09 für den Einstieg in professionelles DTP darstellt, ist „SL" voll echtfarbenfähig und kann modulartig um weitere Funktionen erweitert werden. Beispielsweise zu SLC mit Farbsystemeinbindungen, unter anderem Vierfarbseparation. Ähnlich umlagert waren jedoch auch die Präsentationen bei 3K-ComputerBild, wo das neue Retouche Professional, ein Lithografiesystem mit umfangreichen Bildbearbeitungsfunktionen, auf der Basis des ST gezeigt wurde. Am Stand von MATRIX bzw. IPS schließlich wurden Calamus und Retouche nicht auf dem Atari-Computer, sondern auf einer – wegen besserer Ausbaufähigkeit – selbst entwickelten GEM-Workstation vorgeführt.
Nicht unerwähnt bleiben sollen die Präsentationen des Atari-Computerclubs sowie die Workshops. Auch sie lockten sicher eine Vielzahl der über 42 000 Besucher an, viele erstmals auch aus der (damaligen) DDR.

H. Weiß

Novell mit neuer Version von NetWare 386

Die Novell Inc., Provo/Utah – führender Anbieter von Netzwerkprodukten – kündigte Ende Mai '90 die Version 3.1 des Serverbetriebssystems NetWare 386 an. Die neue Version kombiniert die Vorzüge des 32-Bit-Echtzeitverarbeitungs-Betriebssystems, der Fehlertoleranz und der Sicherheit aus Version 3.0 und bietet darüber hinaus eine Reihe von Erweiterungen, die die Netzwerk-Performance steigern und die Bedienung vereinfachen.
Das Release NetWare 386 v3.1 verfügt über umfassende Erweiterungen. So schaffen standardisierte Programmierschnittstellen (APIs) eine offene Entwicklungsplattform für Programmierer. Eine ANSI-C-Umgebung mit POSIX-Interface ermöglicht das Programmieren in der 32-Bit-Welt, ähnlich wie unter UNIX. Das neue *Enhanced File System* schafft Geschwindigkeitsvorteile von bis zu 30 Prozent beim Zugriff auf die Festplatte. Es unterstützt die neuen externen Speichermedien CD-ROM und WORM ebenso wie die SCSI-Controller der IBM PS/2-Modelle. Es ist möglich, alle Netzwerkdienste und Ressourcen, beispielsweise LANs und Festplattenspeicher, von jedem Netzwerk-Management-System aus zu kontrollieren. Das *Remote Console Utility* erlaubt den Systemadministratoren, alle Netzwerkdienste zentral von ihren Arbeitsplatzrechnern aus zu steuern.
Die Version 3.1 unterstützt eine Vielzahl unterschiedlicher Kommunikationsprotokolle zur Verbindung von unterschiedlichen Systemen für verteilte Anwendungen. Die erweiterte Architektur unterstützt Industriestandard-PCs (SPX, TLI und Sockets) für verteilte Anwendungen. Die Implementierung des *Open Datalink Interface* (ODI) erlaubt es, bis zu 64 LAN-Adapter zu integrieren. Es ist möglich, über einen einzigen Netzwerkadapter im Server mehrere Protokoll-Stacks gleichzeitig zu verarbeiten. So kann beispielsweise ein Ethernet-Adapter gleichzeitig beide Protokoll-Frames – Ethernet 802.3 und Ethernet II – verarbeiten.
Durch die Unterstützung der EISA-Richtlinien erlaubt NetWare 386 v3.1 die Nutzung von mehr als 16 MByte Hauptspeicher, größeren Festplatten und einem erweiterten Cache-Bereich. Dies bringt erhebliche Geschwindigkeitsvorteile, zudem können mehr NLMs (NetWare Loadable Modules) gleichzeitig ablaufen. NetWare 386 v3.1 ist kompatibel zu mehr als 300 Komponenten wie Server, Schnittstellen-Karten, Festplatten und Datensicherungsprodukten. Novells IMSP (Independent Manufactures Support Program) testet die Kompatibilität der Hardware zur NetWare-Umgebung. Novell ist nach eigenen Angaben der einzige Netzwerkhersteller, der den Anwendern ein Programm bietet, welches über die Eignung von Komponenten für ein zuverlässiges Netzwerksystem informiert.
Das neue Release 3.1 sichert auch einen problemlosen Umstieg von NetWare v2.0a, 2.1 und v3.0. Das Upgrade-Programm wurde vereinfacht und arbeitet mit allen NetWare-Versionen ab v2.0 oder höher. Anwender, die bisher NetWare 386 Version 3.0 einsetzten, müssen nur die neuen Update-Dateien kopieren.

Hardware komprimiert Daten

Zur Datenkompression bzw. Dekomprimierung für die Arbeit mit Festplatten oder Disketten gibt es von der Firma Infochip jetzt eine Zusatzkarte mit dem dafür notwendigen Koprozessorschaltkreis. Dieser kann jedoch auch direkt auf dem Motherboard des PCs untergebracht werden. Die Steckkarte halber Länge, *Expanz!* genannt, kann an XTs und ATs ab MS-DOS 3.x genutzt werden.

Erweiterter Speicher für Sharp IQ

Für den überwiegend im gehobenen Manager- und Geschäftsbereich erfolgreich eingesetzten IQ-Electronic Organizer von Sharp – über sieben Millionen Geräte sollen bereits verkauft worden sein – gibt es jetzt eine weitere Speichererweiterung. Mit der 128 KByte fassenden *Gold Card* der Flensburger AMS GmbH anstelle der bisherigen 32- und 64-KByte Cards stehen nun nicht nur Termine, Telefonnummern, Jahrestage oder Memos zur Verfügung, sondern über das Linkprogramm IQ-PROFESSIONAL auch das Users Dictionary bzw. die Wörtertabelle. Über einen PC XT oder AT können die Wörter auch geändert, ergänzt, gelöscht und ausgedruckt werden.

VESA nun mit 72 Hertz

Auf einen neuen Grafikstandard einigten sich Vertreter der VESA-Gruppe, der führende Monitor- und Grafikkartenhersteller angehören, anläßlich der Comdex Spring '90 Anfang Juni in Atlanta. Die Grafikkartenhersteller vereinbarten ein einheitliches Register für VGA Plus und Super-VGA (vgl. MP 3/90, S. 95). Der Standard beinhaltet eine ergonomische Bildfrequenz von 72 Hz sowie die bereits bekannte Auflösung von 800 × 600 Bildpunkten. Daraufhin brachten als erste Hersteller Sigma Designs und Genoa Karten nach den neuen Standards auf den Markt.
Die 16-Bit-Karte *VGA Legend* von Sigma unterstützt nicht nur alle bekannten Grafikstandards, sondern über den neuen VESA-Standard hinaus auch noch eine Auflösung von 1024 × 768 Bildpunkten – diese allerdings nur noch mit 60 Hz. Je nach Speicherausbau (512 KByte oder 1 MByte) können 16 bzw. 256 Farben gleichzeitig dargesellt werden. Im Lieferumfang für diese Karte sind unter anderem Treiber für Autocad, Wordperfect, Word, Wordstar, Lotus 1-2-3, Windows, GEM und Ventura enthalten. Die Version mit 512 KByte RAM kostet rund 1000 Mark.
Auch die *Genoa 6400 A* erreicht eine maximale Auflösung von 1024 × 768 Bildpunkten bei 16 Farben – im Gegensatz zur VGA Legend aber mit 70 Hz. Die 16-Bit-Karte enthält jedoch nur 512 KByte RAM. Sie wird ebenfalls mit einem umfangreichen Treiberangebot geliefert. Ihr Preis liegt bei etwas über 800 Mark.

Computer ohne Tastatur

Zahlreiche Firmen arbeiten an der Entwicklung tragbarer PCs, die ohne Tastatur auskommen. Zu ihnen gehören unter anderem die japanische Firma Sony und die Firma Parcon.
Der von Sony entwickelte Laptop mit der Bezeichnung **Palmtop PTC-500** stellt, nach Ansicht des Unternehmens, gegenüber den gegenwärtig auf dem Markt angebotenen Laptop-Computern eine echte Alternative dar. Sein Vorteil besteht darin, daß er ohne Keyboard arbeitet und damit als sehr anwenderfreundlich gilt. Beim PCT-500 schreibt der Benutzer mit einem elektrischen Stift auf einen Flüssigkristall-Bildschirm jede Art von Befehlen in lateinischen Buchstaben, in chinesischen oder in japanischen Schriftzeichen, die der Computer weiterverarbeiten kann. Mit diesem Laptop sollen vor allem ältere Menschen angesprochen werden, die sich bisher nicht zutrauten, einen PC mit Tastatur zu bedienen. Zur Erkennung der handgeschriebenen Schriftzeichen, wobei das Gerät durch Fuzzy-Logik-Software auch relativ ungenaue Handschriften analysiert, nutzt der Laptop ein sogenanntes *capacitance-sensitive graphics tablet*, das wie ein Sensor unter dem Flüssigkristall-Bildschirm reagiert. Das bedeutet, reagiert die Spitze des Plastikstifts mit dem Bildschirm, so werden die Bewegungen auf dem Bildschirm in elektronische Signale umgewandelt. Die Signale werden gleichzeitig an die CPU weitergeleitet. Stellt die CPU fest, daß die Signale mit einem Muster im internen Wörterbuch des Computers übereinstimmen, dann sucht sie das Schriftzeichen und zeigt es auf dem Bildschirm in wenigen Sekunden an. Der PCT-500 wiegt 1,3 Kilogramm.
Der tragbare **PC der Firma Parcon** erkennt und verarbeitet ebenfalls handschriftliche Eingaben digital. Er ist MS-DOS-kompatibel und nicht größer als ein Bildband. Die Bedienung erfolgt wie beim PCT-500 über ein Display und einen Elektronikstift (Pencil). Der PC sucht sich aus den optisch dargestellten Aufgabenstellungen das gewünschte Programm. Auf die Oberfläche des Displays werden wie in einem Formularbogen in Stichwort-Feldern Notizen geschrieben, die vom Computer erkannt und umgesetzt werden. Aber auch Zeichnungen im DIN A4-Format sollen erkannt und umgesetzt werden können.
Auf dem LC-Bildschirm (640 × 400 Pixel) ist eine dünne Glasplatte (Screen-Overlay) positioniert, unter der sich eine elektrisch leitende Schicht befindet, die die Berührung mit der Messingspitze des Pencils erkennt und umsetzt. Nach der handschriftlichen Eingabe können Zeichnungen und Notizen ausgedruckt und digital weiterverarbeitet werden. *Fa*

Neu von Digital: VAX 4000 Modell 300

Auf der diesjährigen DECWORLD in Boston zeigte Digital Equipment (die Firma verwendet übrigens seit kurzem statt der eingeführten Abkürzung DEC vorzugsweise das Kürzel Digital) unter anderem als Neuerung einen Rechner, der auf Client-Server-Konzepte ausgelegt ist und damit den Minis direkt Konkurrenz macht. Die VAX 4000 Modell 300 enthält unter anderem drei Prozessoren auf RISC-Basis (von der Firma MIPS), die vor allem die Kapazität der Platten-Eingabe/-Ausgabe erhöhen und den Datentransfer via Ethernet unterstützen. Damit wird die CPU entlastet und die Leistung des Gesamtsystems erhöht (etwa 8 VUP, das heißt die achtfache Leistung einer VAX 11/780). Dieses Design macht die VAX 4000 nach Aussagen von Digital zum idealen Server-System, da im Client/Server-Modell der Server besonders hohe I/O-Anforderungen erfüllen muß. Trotzdem bleibt die VAX/VMS-Softwarekompatibilität gewährleistet. Investitionen in bestehende Software sind somit geschützt.
Die VAX 4000 ersetzt die Systeme MicroVAX 3800 und 3900 – mit doppelter CPU-Leistung, mehr als dreifacher Speicher-Bandbreite, einer Leistung von bis zu 22 Transaktionen pro Sekunde –, zu einem niedrigeren Preis. Eine Einstiegsversion für bis zu 40 Benutzer kostet etwa 220000 DM. In der Server-Version ist das System ab 154000 DM erhältlich. Die VAX 4000 kann in Schritten von 32 MByte mit einem Hauptspeicher von bis zu 128 MByte ausgestattet werden. Die Plattenkapazität liegt bei maximal 28 Gigabyte. Es stehen zwölf Steckplätze zur Verfügung, von denen für das System und den Standardspeicher fünf belegt sind.

AS/400 erweitert

Die IBM Deutschland kündigte für die Systemfamilie IBM AS/400 die Modelle CO4 und CO6 sowie für die Systemfamilie IBM AS/Entry das Modell Y10 an. Die neuen Modelle CO4 und CO6 sind für kleinere Unternehmen, Zweigstellen und Abteilungen gedacht und ermöglichen einen günstigeren Einstieg als bisher. Mit der Ankündigung des neuen AS/Entry-Modells Y10 erfüllt IBM die Absicht, die AS/400 und die AS/Entry-Serie noch näher zusammenzuführen. Das Modell AS/400 CO4 ist in der Grundausstattung für 27700,– DM, das Modell AS/400 CO6 für 38600,– DM und das Modell Y10 für 18670,– DM (jeweils plus MwSt.) erhältlich.

*	FONTRESET	Font u. Paletten zurücksetzen
*	EGAPALETTE	Farben unter EGA ändern
*	VGAPALETTE	Farben unter VGA ändern
*	STANDARD	Standard-Farbwert von SET COLOR TO für Ausgaben wählen
*	UNSELECTED	Farbwert f. ungewählte Eingabefehler
*	ENHANCED	Farbwert f. das gewählte Eingabefeld

Disketten- und Festplattenfunktionen

	NUMDISKL	Zahl der logischen Laufwerke
	NUMDISKF	Zahl der Diskettenlaufwerke
*	NUMDISKH	Zahl der Festplatten
*	DISKSPEED	Laufwerks-Geschwindigkeit
	DRIVETYPE	Laufwerks-Typ
	DISKTYPE	Typ einer Diskette
	DISKTOTAL	gesamte Speicherkapazität einer Diskette
	DISKFREE	freier Speicherplatz auf Disk./Platte
	DISKREADY	Ist ein Laufwerk bereit?
	DISKREADYW	Ist ein Laufwerk beschreibbar?
	DISKSTAT	Status des angegebenen Laufwerks
	DISKCHECK	Disketten-Prüfsumme bilden
	FILECHECK	Datei-Prüfsumme bilden
*	FILEVALID	Ist ein String ein gültiger Dateiname?
	TEMPFILE	temporären Dateinamen bilden
*	FILESEEK	Datei nach Namen u. Attribut suchen
*	FILEATTR	Attribute einer Datei ermitteln
*	FILEDATE	Datum der letzten Änderung einer Datei
*	FILETIME	Uhrzeit d. letzten Änderung einer Datei
*	FILESIZE	Größe einer Datei
*	SAVEFSEEK	Fileseek-Environment retten
*	RESTFSEEK	Fileseek-Environment wiederherstellen
*	SETFDATI	Dat. u. Uhrzeit d. letzten Änderung einer Datei
*	SETFATTR	Attribute einer Datei setzen
*	GETSHARE	Dateiöffnungsmodus ermitteln
*	SETSHARE	Dateiöffnungsmodus für Toolbox-Dateifunktionen setzen
*	VOLUME	Volume-Label einer Disk./Platte setzen
*	VOLSERIAL	DOS-4.x-Disketten-Seriennr. ermitteln
	FILEDELETE	Datei(en) nach Namen u. Attribut löschen
	DELETEFILE	Datei fehlertolerant löschen (für Netze, liefert Fehlerkode)
*	FILECOPY	Datei kopieren (auch im Backup-Modus)
*	FILECOPEN	Ist Datei im Backup-Modus noch offen?
*	FILECCLOSE	Datei nach Backup-Modus schließen
*	FILECCONT	Datei im Backup-Modus weiter kopieren
*	FILECDATI	Original- o. Systemzeit f. Kopie setzen?
*	FILEAPPEND	an eine Datei anhängen (Für FILECOPY()-Backups)
	FILEMOVE	Datei in ein anderes Verz. *verschieben*
*	RENAMEFILE	Datei fehlertolerant umbennen
	DIRMAKE	Verzeichnis anlegen
	DIRREMOVE	Verzeichnis löschen
	DIRCHANGE	Verzeichnis wechseln
	DIRNAME	Name des angemeldeten Verzeichnisses
	DISKNAME	Laufwerksbuchstabe ermitteln
*	DISKCHANGE	Laufwerk wechseln
*	STRFILE	String in Datei schreiben
*	FILESTR	String aus Datei lesen

Druckerfunktionen

	NUMPRINTER	Zahl der parallelen Schnittstellen
	PRINTREADY	Ist der Drucker bereit?
	PRINTINIT	Drucker initialisieren
	PRINTSTAT	Status einer parall. Schnittst. ermitteln
	PRINTSEND	Zeichen drucken
	PRINTFILE	Datei mit HiBit, CR und LF drucken
*	PRINTSCR	Bildsch.inhalt drucken (Print-screen)
*	PRINTSCRX	Bildsch.-Druck: nicht druckbare Zeichen austauschen
	TOF	auf Seitenanfang prüfen
*	SPOOLACTIV	DOS-Print installiert?
*	SPOOLADD	Datei zur Druck-Warteschlange addieren
*	SPOOLDEL	Datei aus Druck-Warteschlange löschen
*	SPOOLFLUSH	Druck-Warteschlange löschen
*	SPOOLENTRY	Datei aus Druck-Warteschlange anzeigen bzw. Druck anhalten
*	SPOOLCOUNT	Zahl d. Dateien in Druck-Warteschlange

Datums- und Zeitfunktionen

	DOY	Tag des Jahres
	WEEK	Woche des Jahres
	QUARTER	Quartal des Jahres
	ISLEAP	Schaltjahrbestimmung
	LASTDAYOM	letzter Tag des Monats
	WOM	Woche des Monats
	ADDMONTH	Monat(e) zu einem Datum addieren
	BOM	Monatsbeginn
	EOM	Monatsende
	BOQ	Quartalsbeginn
	EOQ	Quartalsende
	BOY	Jahresbeginn
	EOY	Jahresende
	TIMEVALID	Ist eine Zeitangabe gültig?
	TIMETOSEC	Zeit in Sekunden umwandeln
	SECTOTIME	Sekunden in Zeitstring umwandeln
*	CTODOW	Wochentagsnamen in Nummer umwandeln
*	CTOMONTH	Monatsnamen in Nummer umwandeln
*	NTOCDOW	Wochentagsnummer in Namen umwandeln
*	NTOCMONTH	Monatsnummer in Namen umwandeln
*	DMY	String im Format Tag, Monatsname, Jahr
*	MDY	String im Format Monatsname, Tag, Jahr
	SHOWTIME	Zeit kontinuierlich anzeigen
	STOD	String in Datum umwandeln
	SETDATE	das System-Datum setzen
	SETTIME	die System-Zeit setzen

*	WAITPERIOD	Zeitsteuerung von Schleifen

Datenbankfunktionen

	ISDBT	Sind DBT-Dateien f. Memofelder vorhanden?
	DBFSIZE	Größe der offenen DFB-Datei (Speicher)
	DBFDSKSIZE	Größe der offenen DBF-Datei (Laufwerk)
	FIELDTYPE	Datentyp eines Feldes
	FIELDSIZE	Größe eines Feldes
	FIELDDECI	Zahl der Dezimalstellen eines Feldes
	FIELDNUM	Feldnummer zu einem Feldnamen

Schalter- und Statusfunktionen

	CSETALL	Einstellungen aller On/Off-Schalter retten
	CSETxxxx	Set-Einstellg. f. alle On/Off-Schalter
*	CSETCLIP	Werte der Umgebungsvariablen Clipper
	CSETDATE	landesspez. Datumsversion abfragen
	CSETDECI	Anz. d. eingestellten Dezimalstellen
	CSETDEFA	Einstellg. von SET DEFAULT abfragen
	CSETFUNC	Einstellg. der Funktionstasten abfragen
	CSETKEY	Keytrap-Prozedur, f. eine Taste
	CSETRDEL	Zeichen f. re. Begrenzg. d. Eing.feldes
	CSETLDEL	Zeichen f. li. Begrenzg. d. Eing.feldes
	CSETMARG	Einstellung des linken Druckrandes
	CSETPATH	aktueller Suchpfad
	CSETSNOW	Schnee auf Bildschirm vermeiden
	CSETSAFETY	Dateien vor Überschreiben schützen
*	CSETRDONLY	Zugriffsmodus f. DBF-Dateien best. (R/O, R/W)
	ISDEBUG	Prüf, ob Debugger eingebunden ist
	LASTKLINE	Zeilennr. bei letztem Key-Ereignis
	LASTKPROC	PROCNAME() bei letztem Key-Ereignis
	LASTKFUNC	Funkt., d. letztes Key-Ereignis verurs.
	NUMFKEY	Anzahl der Funktionstasten ermitteln
	KSETINS	Einfügemodus abfragen/setzen
	KSETCAPS	Caps-lock abfragen/setzen
	KSETNUM	Num-lock abfragen/setzen
	KSETSCROLL	Scroll-lock abfragen/setzen
	SETLASTKEY	den Inhalt von Lastkey setzen

Systemfunktionen

*	MEMSIZE	System-Speichergröße
	BOOTCOLD	Kaltstart durchführen
	BOOTWARM	Warmstart durchführen
	ISMATH	Ist mathem. Koprozessor vorhanden?
	ISAT	Ist der Computer ein AT?
*	PCTYPE	Computer-Typ ermitteln
*	CPUTYPE	Prozessor-Typ ermitteln
	OSVER	DOS-Versionsnummer ermitteln
	SSETVERIFV	(DOS-)VERIFY prüfen/setzen
	SSETBREAK	(DOS-)-BREAK prüfen/setzen
	EXENAME	Verzeichnis u. Name d. akt. Clipper-Progr.
	DOSPARAM	DOS-Kommando-Zeile komplett übernehmen
*	ENVPARAM	ges. Environment in String einlesen
	GETCOUNTRY	landesspez. Einstellung d. Betriebssyst.
	ERRORCODE	Kennziffer eines vorherigen DOS-Fehlers
	ERRORBASE	Ursche f. zuvor aufgetret. DOS-F.
	ERRORACT	empfohlene Aktion f. zuvor aufgetr. DOS-F.
	ERRORORG	Fehler-Quelle f. zuvor aufgetr. DOS-F.
*	ISANSI	ANSI-Bildschirm-Treiber installiert?
*	BIOSDATE	Datum des Computer-BIOS ermitteln
*	NUMBUFFERS	Einstellung für *Buffers* ermitteln
*	NUMFILES	liefert Einstellung für *Files* (liest aber nicht die Datei Config.sys)
*	FILESMAX	Wieviele Dat. können max. geöffn. werden?
*	FILESFREE	Wieviele Dat. können noch geöffn. werden?

Netzwerkfunktionen (PC-LAN/MS-NET)

*	NETWORK	Ist ein Netzwerk aktiv?
*	NETDISK	Laufwerk lokal o. auf Server (PC-LAN/Novell)?
*	NETPRINTER	Drucker lokal o. auf Server (PC-LAN)?
*	NETREDIR	lokale Einheit auf Server-Einh. umlenken
*	NETCANCEL	Umlenkung lokaler Einheit aufheben
*	NETLOCNAME	Name einer lokalen, umgelenkten Einheit
*	NETRMTNAME	Name d. Server-Einh. zu lokal umgelenkter Einh.
*	NNETWORK	Ist Novell-Netzwerk aktiv?
*	NNETSPSTAT	Status eines Server-Druckers
*	NNETCAPBEG	Umlenkung auf bestimmten Drucker unter Novell (Capture-Modus)
*	NNETCAPSSF	Einstellung für Capture-Modus
*	NNETCAPACT	Ist ein Capture aktiv?
*	NNETCAPCAN	Umlenkung LPTn aufheben, momentanen Job löschen
*	NNETCAPEND	Umlenkung LPTn beenden
*	NNETCAPFLU	momentanen Druckjob schließen, abschicken
*	NNETSETQ	Drucker-Warteschlange f. Capture-Modus anmelden
*	NNETLOGGED	Ist diese Station eingeloggt?
*	NNETSTAID	Identifik. d. Station (Netzwerk-Karte)
*	NNETCOPRIV	Nutzer berechtigt, Server zu bedienen?
*	NNETWHOAMI	Unter welchem Namen bin ich angemeldet?
*	NNETVER	Vers. d. verwend. Novell-Netzwerk-Softw.
*	NNETINFO	Info zur verwend. Novell-Software
*	NNETSFTLVL	SFT-Level d. verwend. Novell-Netzwerk-Softw.
*	NNETSDATE	Datum des Servers
*	NNETSTIME	Uhrzeit des Servers
*	NNETINUSE	Wieviele Personen (Objekte) sind momentan angemeldet?
*	NNETOCNUMS	liefert alle Stationen, auf denen ein User angemeldet ist
*	NNETUSERID	Identifik. eines Nutzers innerh. d. Netzes
*	NNETUSRFRE	die für einen Nutzer noch verfügbare Server-Plattenkapazität
*	NNETPURGE	alle gelöschten Dateien tatsächlich entfernen
*	NNETSNAME	Namen d. Standard-File-Servers ermitteln
*	NNETSLIST	alle angemeld. Server in einem String
*	NNETSETSRV	angemeld. Server als Standard-Server best.
*	NNETVOLNUM	Nr. d. höchsten, verfügbaren Volume
*	NNETVOLNAM	Name einer Volume auf dem Server
*	NNETVOLMAX	max. Kapazität einer Server-Volume
*	NNETVOLFRE	freie Kapazität einer Server-Volume
*	NNETDIRMAX	max. Anz. von Directory-Einträgen einer Volume
*	NNETDIRFRE	Anz. freier Directroy-Einträge einer Volume
*	NNETERROR	zuletzt bei Novell-Funkt. aufgetr. Fehler

Eingabefunktionen

*	SAVEGETS	alle akt. Get-Einstellungen in einem String retten
*	RESTGETS	Get-Einstellungen aus String zurücksetzen
*	SAVESETKEY	alle SET KEY TO-Einstellungen in einem String retten
*	RESTSETKEY	alle in einem String gesicherten SET KEY TO-Einstellungen zurücksetzen
*	GETFLDVAR	Variable zu einem Get-Feld
*	GETFLDROW	Bildschirmzeile eines Get-Feldes
*	GETFLDCOL	Bildschirmspalte eines Get-Feldes
*	CURRENTGET	Anzahl der aktiven Get-Felder
*	COUNTGETS	Anzahl der aktiven Get-Felder
*	GETINPUT	einem Get-Feld sehr ähnl. Eingabefunkt.
*	GETSECRET	Eingaben wie mit GETINPUT(), jedoch durch "*" verdeckt

Mathematische und kaufmännische Funktionen

*	FV	künftiges Kapital (future value)
*	PV	Barwert einer Verzinsung (present value)
*	PAYMENT	Wie hoch muß periodische Zahlung sein?
*	PERIOD	Über wieviele Perioden muß gezahlt werden?
*	RATE	Wie hoch ist Zinssatz für Kredit?
*	DTORE	Gradmaß in Bogenmaß wandeln
*	RTOD	Bogenmaß in Gradmaß wandeln
*	CEILING	aufrunden
*	FLOOR	abrunden
*	SIGN	das Vorzeichen einer Zahl
*	PI	Zahl π mit größtmöglicher Genauigkeit
*	SETPREC	Rechengenauigkeit für trigonometrische Funktionen einstellen
*	GETPREC	Einstellg. f. Rechengenauigkeit ermitt.
*	SIN	Sinus
*	COS	Kosinus
*	TAN	Tangens
*	COT	Kotangens
*	ASIN	Arcus Sinus
*	ACOS	Arcus Kosinus
*	ATAN	Arcus Tangens
*	ATN2	Winkelgröße aus Sinus u. Kosinus berechn.
*	FACT	Fakultät
*	LOG10	dekadischer Logarithmus

Port- und Speicherzugriff

*	INBYTE	8 Bit vom Port lesen
*	INWORD	16 Bit vom Port lesen
*	OUTBYTE	8 Bit zum Port senden
*	OUTWORD	16 Bit zum Port senden
*	PEEKBYTE	8-Bit-Zahl aus dem Speicher lesen
*	PEEKWORD	16-Bit-Zahl aus dem Speicher lesen
*	PEEKSTR	Speicherbereich als String lesen
*	POKEBYTE	8-Bit-Zahl im Speicher ablegen
*	POKEWORD	16-Bit-Zahl im Speicher ablegen

Verschiedenes

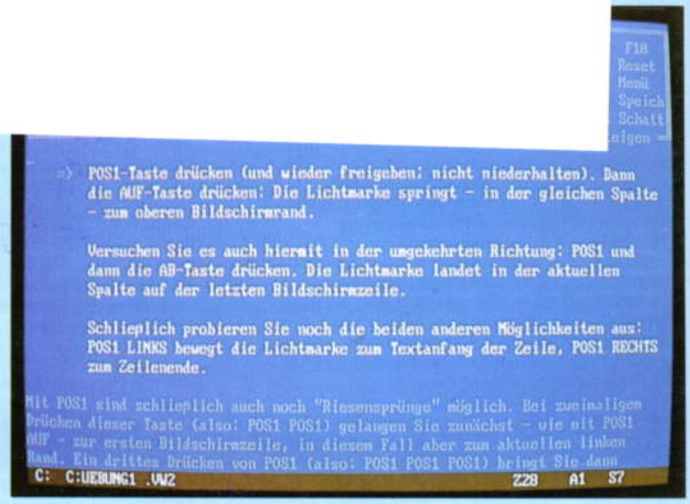
*	SHOWKEY	Einfügemodus u. *LOCK*-Zustände autom. anzeigen (Indikatoren)
	KEYSEC	Keybord-Taste nach best. Zeitspanne in Puffer legen
	KEYTIME	K.-Taste zu best. Uhrzeit in Puffer legen
*	KBDEMULATE	Zeichen im BIOS-Tastatur-Puffer ablegen (Eingabe emulieren)
*	KBDTYPE	Tastatur-Typ ermitteln
*	KBDSPEED	Tastatur-Geschwindigkeit einstellen
*	KBDDISABLE	Tastatur sperren/entsperren
*	KBDSTAT	Tasten-Umschaltmodi. u. Status v. Steuertasten
	SCANKEY	der Scan-Kode einer Eingabe
	SOUND	Ton best. Frequenz u. Dauer erzeugen
	NUL	Rückgabewert einer Funkt. zu Leerstring wandeln
*	TOOLVER	Toolbox-Version ermitteln/Treiber-Library überprüfen
*	MILLISEC	Verzögerung in 1/1000 Sekunden erzeugen
*	SPEED	Prozessor-Geschwindigkeit ermitteln
	XTOC	Daten beliebigen Typs in String umw.
	DATATYPE	Datentyp einer Variablen o. einer nutzer-definierten Funktion
	BLANK	Leerwert f. beliebigen Datentyp erzeugen
	COMPLEMENT	Komplementärwert zu beliebigem Datentyp bilden
*	ALLOFREE	frei verfügbare Hauptspeichergröße ermitteln
*	STACKFREE	den noch freien Stackbereich ermitteln

Volkswriter

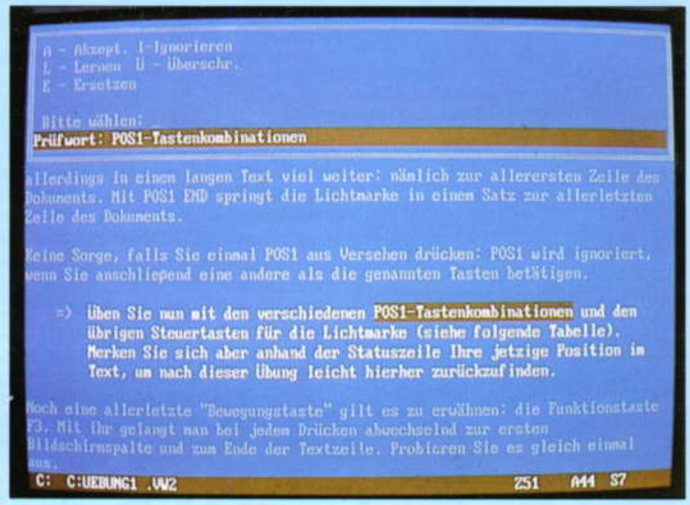
Textverarbeitung modular

Kann man Texte modular verarbeiten? Oder handelt es sich bei Volkswriter vielmehr um ein modular aufgebautes Textverarbeitungsprogramm? Aufschluß darüber gibt der Werbeslogan der gleichnamigen Firma – „Ihr Einstieg in die modulare Textverarbeitung" – jedenfalls nicht. Aber gerade mit dem Modulkonzept von Volkswriter wird Einsteigern der Übergang zur professionellen Textverarbeitung stark erleichtert.
Aus der Erkenntnis heraus, daß 70 % aller Anwender nur etwa 20 % der Leistungsmerkmale von Textprogrammen nutzen, entstand 1988 das kostengünstige Einsteiger-Textsystem Volkswriter deluxe. Weil aber auch Einsteiger eventuell irgendwann höhere Ansprüche stellen, wurde anschließend das modulare Volkswriter-Konzept entwickelt: Aus *deluxe* wurde *1*, und hinzu kamen Volkswriter2 (mit Rechtschreibkontrolle), Volkswriter3 (mit automatischer Silbentrennung und Fremdsprachenmodul) und Volkswriter3 NET (mit Netzbetrieb). Volkswriter3 steht solchen Textprogrammen wie Word und Wordstar in nichts nach – auch nicht im Preis; der liegt so zwischen 850 und 950 Mark. Dagegen bekommt man Volkswriter1 schon für runde 140 Mark – wie gesagt, lediglich ohne Silbentrennung und Rechtschreibprüfung.
Bei der Installation von Volkswriter bemerkt man, daß dem Käufer die Arbeit so leicht wie möglich gemacht werden soll, sieht man mal von einigen Widersprüchen ab, die sich bei sehr unüblichen Installationsvarianten ergeben. Die Handgriffe sind zwar menügeführt, werden aber zudem ausführlich im Handbuch erläutert.
Besonders interessant dürfte Volkswriter1 für Besitzer von Rechnern ohne Festplatte oder mit einem kleinen Hauptspeicher sein, da die meisten anderen Textverarbeitungsprogramme wegen ihrer Länge mehr als 256 KByte RAM und eine Festplatte voraussetzen. Eine Installation auf einem XT mit 256 KByte Hauptspeicher, 2 Floppylaufwerken und MS-DOS 2.11 gelang mühelos. Wollen Sie jedoch Volkswriter1 gleich nach dem Einschalten des Rechners durch die Autoexec.bat von der Boot-Diskette aus starten lassen, dann müssen Sie vorher einige selten benötigte externe DOS-Kommandos von der 360-KByte-Diskette räumen. Volkswriter2 dagegen ist mit seinem über 1 Million Wörter umfassenden Rechtschreibmodul erst auf Festplattensystemen sinnvoll einsetzbar.
Auch die Druckeransteuerung – das leidigste Thema bei Textprogrammen – bereitet in den meisten Fällen keine Schwierigkeiten, da Sie sich Ihren Drucker aus einer Palette von rund 400 aussuchen können; außerdem wird zusätzlich ein Druckerhandbuch mitgeliefert. Sollten Sie ihren Drucker in der Aufstellung jedoch nicht finden, oder druckt er einige Sonderzeichen trotzdem nicht oder falsch, dann erklärt Ihnen das Handbuch, wie Sie sich leicht eine Drucker-Übersetzungstabelle anlegen können.
Etwas schwieriger (wegen des Heraussuchens der Buchstabenkodes aus verschiedenen Tabellen) ist dagegen das Anlegen einer Übersetzungstabelle für die Tastatur, zum Beispiel um häufig benutzte Sonderzeichen mit einer beliebigen Buchstaben- oder Zifferntaste erreichen zu können – nach Bedarf auch mit einer <Alt>-Tastenkombination. Damit ist dann beispielsweise auch das Schreiben in einer anderen Sprache oder das „Malen" eines Rahmens kein Problem. Komfortabel ist auch die Arbeit mit sogenannten Layouts und Formblättern. So können Sie zum Beispiel Ränder, Tabulatoren und Zeilenabstände innerhalb eines Textes mehrfach ändern, indem Sie einfach ein anderes Layout anwählen. Wollen Sie aber das gerade benutzte Layout speichern, dann rufen Sie im Hauptmenü den Punkt Formblätter auf. Dort legen Sie dann fest, ob dieses Layout nun (als Formblatt) für bestimmte oder alle Dateitypen gelten soll; das bewirkt, daß nun mit dem Laden einer Datei das jeweils gültige Formblatt mitgeladen wird. Damit können Sie beispielsweise mit dem Dateityp *.bri* Briefe immer einzeilig und mit *.txt* Manuskripte für die MP zweizeilig schreiben.
Angenehm ist mir auch die Einhand-Kursorbedienung aufgefallen: Mit dem aufeinanderfolgenden – nicht gleichzeitigen – Drücken der Tasten <Pos1> (Home) sowie einer Kursortaste, <End> oder wieder <Pos1> sind große Kursorsprünge möglich. Nach einer kurzen Einarbeitung ist diese Kursorbedienung schneller als eine Zweihand-Bedienung oder die Arbeit mit Pull-down-Menüs.
Nun zur Rechtschreibprüfung: Um festzustellen, welche von mir benutzten Wörter der Duden von Volkswriter2 noch nicht kennt, habe ich den Text dieser MP-Seite prüfen lassen. Dabei waren neben einigen längeren zusammengesetzten Wörtern unter anderem folgende Wörter für Volkswriter noch unbekannt: Einsteiger, Netzbetrieb, Hauptspeicher, Duden und sogar Volkswriter1 bis 3. Bei der Prüfung wählte ich jeweils die Funktion *Lernen*, damit die Wörter in den Duden aufgenommen werden. Ein zweiter Lauf, bei dem die Rechtschreibprüfung ohne Halt durchlief, ergab mit einem 8-MHz-AT eine Prüfzeit von 22 s für rund 700 Wörter.
Wer in seine Texte auch Daten übernehmen will, der kann (optional) von Volkswriter das Programm Slice nutzen. Es ermöglicht auf einfache Weise die Datenübernahme aus dBase- oder Lotus 1-2-3-Dateien (für über 50 verschiedene Textprogramme). Die übernommenen Daten können danach mit der Rechenfunktion in Volkswriter3 bequem zu einer Statistik aufbereitet werden.

H. Hemke



Textbearbeitung – alle Bearbeitungsfunktionen können mit den vierfach belegten Funktionstasten aufgerufen werden



Rechtsschreibprüfung – fehlerhafte, doppelte oder unbekannte Wörter werden markiert

Einige Leistungsmerkmale

Volkswriter-Version	1	2	3	3NET
Voraussetzungen				
DOS ab Version	2.0	2.0	2.0	3.1
RAM	256 KByte	320 KByte	384 KByte	384 KByte
Festplatte	nein	ja	ja	ja
Leistungsmerkmale				
autom. Silbentrennung	nein	nein	ja	ja
Festlegen d. Trennzone	nein	nein	ja	ja
Rechtschreibprüfung	nein	ja	ja	ja
Synonymwörterbuch	nein	optional[1]	optional[1]	optional[1]
Mehrsprachenunterstützung	nein	nein	ja	ja
Rechnen im Text	nein	nein	ja	ja
Sortierfunktion	nein	nein	ja	ja
Layoutfunktion	ja	ja	ja	ja
netzfähig	nein	nein	nein	ja
mehrplatzfähig	nein	ja[2]	ja[2]	ja[2]
Dokumentenlänge	1 MByte	1 MByte	1 MByte	1 MByte
Mausunterstützung	optional	optional	optional	optional
Hintergrunddruck	ja	ja	ja	ja
Zahl der Druckertreiber	ca. 400	ca. 400	ca. 400	ca. 400
ASCII-Dateien	ja	ja	ja	ja
Preis	139 DM	279 DM	877 DM[3]	1276 DM[4]
Upgradepreis	–	–	319 DM (Von Version 1 nach 3: 399 DM)	319 DM

Alle Preise inklusive MwSt.
[1] Ein Synonymwörterbuch ist für 99 DM erhältlich.
[2] in Verbindung mit PC-MOS 4.0
[3] Im Preis ist ein Modul für 1 Fremdsprache enthalten, jedes weitere kostet 99 DM.
[4] Im Preis ist ein Modul für 1 Fremdsprache enthalten. Die Netzversion arbeitet mit 3 Usern, jedes weitere User-Modul kostet 189 DM.

ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

MP

Heft 12 · 1990

Mikroprozessortechnik

Verlag Technik Berlin
ISSN 0232-2892
DM 5,-

Leiterplatten-entwicklung am PC

WordPerfect 5.1

WordPerfect liegt eine andere Idee zugrunde, als dem weit verbreiteten Wordstar. Statt ASCII-Steuerzeichen mit <Ctrl> und einem Buchstaben sowie Punktkommandos im Quelltext werden ausschließlich die (mit <Shift>, <Ctrl> und <Alt>) vierfach belegten Funktionstasten <F1> bis <F10> genutzt. Steuerzeichen stehen unsichtbar im Quelltext, es sei denn, man schaltet sie mit <Alt> <F3> zur Kontrolle sichtbar. So kann man die in eckigen Klammern dargestellten, gut verständlichen Kürzel auch leicht von Hand korrigieren. Leider wurden einige unübliche Tastenkombinationen gewählt, etwa <ESC> zur Einstellung eines Wiederholfaktors. Die üblicherweise dort gesuchte Stornofunktion ist über <F1> erreichbar, wo man eigentlich Hilfe erwartet, die aber auf der <F3>-Taste liegt. Hat man sich erst einmal an die Tastenbelegungen gewöhnt (was zudem durch eine mitgelieferte Schablone unterstützt wird), läßt sich mit WordPerfect flott arbeiten.

Die auffälligste Neuheit der Version 5.1 (die uns bereits als deutsche Version vorlag) ist die Menüführung mittels Pulldown-Menüs, zu deren Bedienung jetzt auch die Maus verwendet werden kann. Daneben stehen aber die traditionellen *Shortcuts* für den fortgeschrittenen Nutzer weiterhin zur Verfügung, und wer die Menüs nicht benötigt, kann sie auch ausblenden. Es ist augenfällig, daß die neue Nutzeroberfläche nur aufgesetzt wurde, denn nach Auswahl eines Punktes befindet man sich wieder in den alten Menübildschirmen, was aber den Vorteil einer gewohnten Umgebung für den erfahrenen Benutzer mit sich bringt. In einer Zeit, da immer mehr Programme mit SAA-konformen Dialogboxen und Scroll-Menüs aufwarten, mutet die Auswahl der Menüpunkte durch Ziffern aber nicht sehr modern an. WordPerfect

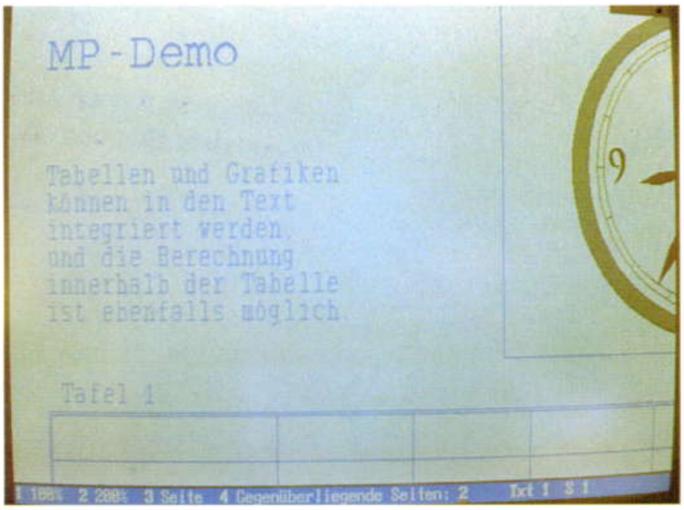


Bild 2

erlaubt zwar die gleichzeitige Bearbeitung von zwei Texten, verzichtet aber auf die Fenstertechnik. Zur besseren Übersicht ist eine farblich differenzierte Darstellung beider Texte möglich. Positiv fällt die neue Verwaltung der Texte durch WordPerfect auf. Statt der bei MS-DOS auf 8 Zeichen begrenzten Dateinamen kann man jetzt bis zu 40 Zeichen (einschließlich Leerzeichen) angeben, etwa *Brief an Susi*. Komfort, der an den Macintosh erinnert. Weiterhin ist das Eintragen des Bearbeiters, des Texttyps und eines Kommentars möglich. Während des Schreibens wird der Text automatisch formatiert und - sofern eingestellt - auch getrennt. Auf die automatische Trennung sollte man sich aber nicht verlassen. Zur Unterstützung der deutschen Trennregeln steht ein Ausnahmewörterbuch zur Verfügung, das die allgemeinen Regeln ergänzt und das vom Nutzer selbst erweitert werden kann. Wird die Trennung nach Wörterbuch ganz abgeschaltet, so wird nach englischen Trennregeln verfahren.

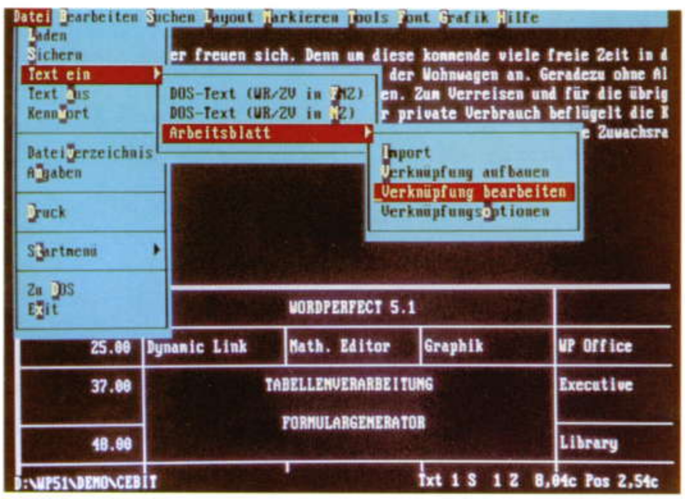


Bild 1

Welcher Abstand vom Rand höchstens erlaubt ist, kann eingestellt werden. Blocksatz macht sich nur beim Druck oder bei der Druckvorschau bemerkbar; während des Editierens wird - im Gegensatz zu Wordstar - nicht mit Leerzeichen aufgefüllt. Etwas verwirrend ist, daß Formatfestlegungen nicht global getroffen werden, sondern stets ab der aktuellen Kursorposition gültig sind und (unsichtbar) in den Quelltext (auch mehrmals) eingetragen werden. Hier geht schnell die Übersicht verloren.

WordPerfect bietet alles, was zu einer modernen Textverarbeitung gehört: Die Verwaltung der Fußnoten und des Anhanges sowie von Index-, Quellen- und Inhaltsverzeichnissen. Zusätzlich gibt es Funktionen, die man sonst nur von DTP-Programmen wie Ventura Publisher und PageMaker kennt, nämlich die Arbeit mit sogenannten Styles. Einmal definierte Schrifttypen können gespeichert und später wieder verwendet werden. Darüber hinaus ist auch die Festlegung ganzer Layouts möglich, in die später nur noch die Texte geladen zu werden brauchen. Auf diese Weise können Publikationen (Zeitschriften, Broschüren) mit geringem Aufwand stets gleich gestaltet werden.

Die beiden Seiten dieses Beitrages zeigen, wie dies im Druck aussieht. Der Text wurde mit WordPerfect erfaßt, formatiert und als PostScript-Emulation über einen Laserdrucker mit einer Auflösung von 300 dpi ausgegeben. Für die in der Druckerei

Fortsetzung siehe 3. Umschlagseite

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
4. Jahrgang · ISSN 0232–2892

Mikroprozessortechnik, Heft 12 · 90

Verlag
Verlag Technik GmbH, Oranienburger Str. 13/14,
O – 1020 Berlin; Telegrammadresse: Technikverlag Berlin;
Telefon: 28700, Telex: 011 2228 techn dd,
Telefax: 2 87 02 59

Geschäftsführer
Klaus Hieronimus

Verlagsleiter Zeitschriften Günter Fitzner

Redaktion
Hans Weiß, Chefredakteur (Tel. 2870371);
Redakteure: Herbert Hemke (Tel. 2 87 02 03),
Hans-Joachim Hill (Tel. 2870209);
Sekretariat: Hannelore Kulik (Tel. 2870381)

Gestaltung
Christina Bauer

Titel
Frank-Norbert Beyer

Gesamtherstellung
Märkische Verlags- und Druck-Gesellschaft mbH Potsdam

Anzeigenannahme
Verlag Technik GmbH, Anzeigenabteilung, Oranienburger Straße 13/14, O–1020 Berlin;
Tel. 2 87 02 54, 2 87 03 09, Telefax: 2 87 02 54

Anzeigenpreise Preisliste Nr. 1 vom 1. Juli 1990

Redaktionsschluß 10. Oktober 1990

AN (EDV) 49837

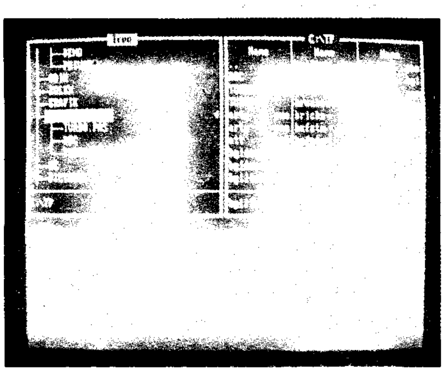
Erscheinungsweise monatlich

Bezugspreise
Einzelheft 5,– DM, Abonnementpreis 15,– DM vierteljährlich incl. MwSt.
Der Abonnementpreis erhöht sich für das Ausland um die Zustellgebühren.
Das Abonnement verlängert sich automatisch um den bestellten Zeitraum, wenn es nicht 6 Wochen vor Ablauf der Frist bei der Bestelladresse schriftlich gekündigt wird.

Bezugsmöglichkeiten
Bestellungen sind an den Verlag zu richten.
Abonnementbestellungen für die westlichen Länder der BRD und das westliche Ausland nimmt entgegen
Inter abo · Betreuungs-GmbH
Wendenstr. 25, Postfach 10 32 45
W-2000 Hamburg 1
Bundesrepublik Deutschland
Interessenten in Österreich übergeben ihre Bestellung der Firma Hartleben
Dr. Rob
Buchhandlungen
Schwarzenbergstraße 6
A-1015 Wien

Zum Titelbild:

Sie müssen ja nicht auch gleich Bleistift und Lineal in den Papierkorb werfen, wenn Sie ihre Leiterplatten künftig an Ihrem „persönlichen Computer" entwickeln wollen. Aber brauchen werden Sie diese Utensilien dann nur noch selten. Denn viele der vertrauten Tätigkeiten kann Ihnen der PC abnehmen. Welche das sind und ob sich die PC-Anschaffung damit auch für Sie lohnt, erfahren Sie in unserem Beitrag auf der **Seite 4**.



Im 5. Teil unseres Kurses zu MS-DOS wollen wir Ihnen eine Möglichkeit aufzeigen, wie Sie sich durch eine komfortable Nutzeroberfläche die ansonsten recht umständliche Bedienung von DOS auf wenige Handgriffe reduzieren können. Der auf der **Seite 19** vorgestellte **Norton Commander** hat aus diesem Grunde auch eine recht weite Verbreitung gefunden. Einige weniger bekannte Detailinformationen sind sicher auch für den schon versierten Anwender interessant.

Das Datenbankentwicklungssystem Clipper bietet leider (noch) keine von vielen anderen Systemen her bekannte integrierte Entwicklungsumgebung, um daraus den Compiler, den Linker und den Debugger aufrufen zu können. Wie Sie dem mit den einfachen Mitteln einer Batch-Datei abhelfen können, zeigt unser Beitrag auf der **Seite 15**.

Vorschau

Für das Heft 1/1991 haben wir für Sie Beiträge zu folgenden Themen vorbereitet:

- Objektorientierte Programmierung
- Clipper 5.0
- PCs mit dem 386sx
- DR DOS 5.0

Inhalt

kennengelernt
WordPerfect 5.1 — **2. US**

MP-Info — **2**

Lutz Mantz:
Leiterplattenentwurf am PC — **4**

Jörg Wunsch:
Schneller Logarithmieralgorithmus für Mikrorechner — **9**

Konrad Schultz:
Grundlagen des logischen Programmierens (Teil 3) — **10**

Bernd Matzke:
Komfortable Funktionsberechnung mit Turbo-Pascal — **12**

Eberhard Klauk:
Eine Batchdatei steuert Clipper — **15**

Wolfgang Schreiber:
Zweites Floppylaufwerk mit 80 Spuren — **16**

kennengelernt — **17**
Uwe Schulze:
Microsoft Quick-Pascal 1.0

Wegbereiter der Informatik
Alan Mathison Turing — **18**

MP-Kurs: — **19**
Wolfram Schulze, Uwe Schulze:
Einführung in MS-DOS (Teil 5)

Carsten Meyer, Bernd Behr:
Connections (Teil 2) — **26**

Rolf Schmieder:
Längenbestimmung von Turbo-Pascal-Overlays — **28**

Wolfgang Franke:
Modulare Tabellensteuerprogramme — **29**

Michael Eiker:
Prüfung der Laufwerke — **31**

Nino Turiansky:
Dateien mit Rückmeldung löschen — **32**

Michael Kretzschmar:
Trigonometrische Funktionen in dBase III — **33**

Christian Hanisch:
Die Interrupts 25H und 26H unter Turbo-Pascal — **34**

Stephan Weinmeister:
Semantische Netze am PC entwickeln — **35**

MP-Computer-Club — **36**
Wolfgang Neumann:
Bildschirminhalte abspeichern
Roland Radandt:
Sortiertes Verzeichnis ausgeben
Walter Fürtig:
Segmentregister

Entwicklungen und Tendenzen — **38**

MP-Literatur — **40**

Der deutsche Workstationmarkt

Nach Aussagen der IDC Deutschland GmbH in ihrer jüngsten Studie „Der deutsche Markt für Workstations 1989–1995“ wurde erstmals die Grenze von 1 Milliarde DM überschritten. Ausgeliefert wurden 1989 24 000 Einheiten, davon etwa 80 Prozent der Geräte für den wissenschaftlichen/technischen Bereich und etwa 20 Prozent für kommerzielle Aufgaben. An der Spitze liegen Hewlett-Packard/Apollo-Systeme mit 44 Prozent Marktanteil, gefolgt von Digital Equipment (26 Prozent) und Sun (18 Prozent). Danach folgen Silicon Graphics, Intergraph, PCS und IBM. Systeme mit RISC-Architektur (Reduced Instruction Set Computers) hätten mit 20 Prozent Anteil gegenüber den herkömmlichen CISC-Systemen (Complex Instruction Set Computers) bisher eine untergeordnete Rolle gespielt. Allerdings erwartet man, daß sich der RISC-Anteil in dem insgesamt stark wachsenden Workstationmarkt deutlich erhöhen wird. Per Anfang 1990 waren in der (damaligen) BRD knapp 58 400 Systeme installiert. Die genannte Studie kann für 4200,– DM (plus MwSt.) bei der IDC Deutschland GmbH, Westerbachstraße 23, W-6242 Kronberg 1, angefordert werden.

EDV-Komplettlösungen

Die Texas Instruments Deutschland GmbH und die Garmhausen & Partner GmbH kooperieren, um auf der Basis von TI-Computern Komplettlösungen zu vermarkten. Das Bonner Software- und Systemhaus Garmhausen & Partner konnte in Ostdeutschland bereits mehrere Geschäftsstellen errichten, wobei man auch auf den Stamm der bei Robotron mit Unix befaßten Mitarbeiter zurückgriff. Denn im Vordergrund der Kooperation stehen Unix-Anwendungen auf der Basis der TI-Mehrplatzrechnerfamilie S 1500 (Motorola-Prozessoren) und von leistungsfähigen PCs auf Basis der Prozessoren 80386 und 80486. TI gewährleistet mit elf Stützpunkten im bisherigen Bundesgebiet, daß im Regelfall spätestens vier Stunden nach einer Schadensmeldung beim Kunden vor Ort für Abhilfe gesorgt wird; dieser Service soll nun auch in Ostdeutschland angeboten werden können.

Neues von Novell

Auch die Novell GmbH, Düsseldorf, bekannter Anbieter von Server-Betriebssystemen, baut ein Netz von Partnern in Ostdeutschland auf. Als erster Schritt wurde mit sechs Unternehmen eine Zusammenarbeit vereinbart. Als Systemhaus beziehungsweise Trainingszentrum (NAEC – Novell Authorized Education Center) fungieren künftig: MEL Industrieautomation GmbH, sfi – Systemhaus für Informationsverarbeitung GmbH, CVU Systemhaus GmbH, CVU-Schulungszentrum GmbH, SRS Software und Systemhaus Dresden sowie Robotron-Bürocenter Dresden GmbH, die seit September als Bull Systemhaus Dresden GmbH firmiert.

Zur Zeit werden die angehenden Novell-Spezialisten in Ost-Berlin und Düsseldorf geschult, die nach Tests dann den Titel CNE (Certified NetWare Engineer) oder CNI (Certified NetWare Instructor) führen. Durch diese Autorisierungsmaßnahmen soll gewährleistet werden, daß Unternehmen in der Ex-DDR das Knowhow besitzen, Support und Schulungsleistungen zu übernehmen und den Anwender qualifiziert bei der Einführung von Netzwerkkomponenten zu unterstützen. Auch die Einführung von Novell-Fachhändlern kann direkt und vor Ort von den neuernannten NAECs übernommen werden.

Mit einem Bündel von Maßnahmen geht die amerikanische Novell Inc. derzeit gemeinsam mit ihren Tochtergesellschaften in Europa gegen Grauimporte vor. Der Anbieter von Server-Betriebssystemen für lokale und unternehmensweite Netzwerke unterstützt damit seine offiziellen Distributionspartner und autorisierten Händler gegen Grauimporteure, die Novell-Produkte ohne adäquaten Support liefern. So gibt Novell dem Handel in Deutschland eine Liste empfohlener Endverkaufspreise an die Hand. Das schafft nach Worten von Roland Richter, Verkaufsleiter für Deutschland und Österreich, „Transparenz im Preisgefüge und ermöglicht allen Marktpartnern eine bessere Orientierung“.

Unisys kooperiert mit CVU

Die Unisys Deutschland GmbH, Sulzbach (Taunus), und die Computer Vertriebs-Union Berlin GmbH, Ost-Berlin, eine der Robotron-Nachfolgegesellschaften, haben ein Kooperationsabkommen unterzeichnet. Danach wird die CVU die hundertprozentige Tochtergesellschaft UNICON gründen, die in Zukunft exklusiv für Unisys Arbeitsplatzrechner und Mehrplatzsysteme in der Ex-DDR vertreiben wird. Im Beirat der UNICON wird Unisys mehrheitlich vertreten sein. Unisys steuert sein Knowhow u. a. durch die Bereitstellung von drei Ausbildern und einem Manager bei und wird auch für die umfangreiche Ausstattung mit Demonstrationsgeräten, Marketing-Materialien, Software usw. sorgen.

Die neue Firma wird zunächst etwa 40 Mitarbeiter haben, dazu kommen noch etwa 10 Servicemitarbeiter aus dem ehemaligen Robotron-Betrieb, die den Servicebereich von Unisys ergänzen. Zur Zeit wird ein flächendeckendes Servicenetz für Unisys-Informationssysteme aufgebaut. Hauptsitz von UNICON ist Berlin, Zweigstellen sind in Chemnitz, Erfurt, Leipzig, Magdeburg und Rostock geplant.

Ingenieurbetrieb gemeinsam mit Siemens

Der Ingenieur-Betrieb Gera (IBG) – ehemals Ingenieurbetrieb für die Anwendung der Mikroelektronik im Bezirk Gera – baut sein Beratungs- und Leistungsangebot in der Industrieautomatisierung durch Kooperation mit Siemens aus. Die IBG mit derzeit 40 Mitarbeitern ist der erste Vertriebspartner für das Siemens Produktgebiet Computer Aided Engineering (CAE) mit der Konstruktionssoftware Sigraph-CAE sowie Simatic und Sicomp/PC für die Fertigungssteuerung. Die IBG wird ihren Kunden – zu denen vor allem Industrieunternehmen gehören – auch die Siemens Produktpalette für die Industrie- und Fertigungsautomation anbieten. Die IBG-Mitarbeiter werden zur Zeit im Umgang mit neuen Rechnergenerationen und deren Einsatz bei integrierten Verfahrenslösungen geschult.

Express-Service für Leistungshalbleiter

Das AEG/Siemens-Unternehmen eupec hat jetzt einen Express-Service eingerichtet, mit dem bei kleinen Stückzahlen die Bauelemente innerhalb von 48 Stunden nach Auftragseingang ausgeliefert werden können. Voraussetzung dafür waren der Einsatz moderner Technologien bei der Abwicklung und Bearbeitung der Bestellung und eine optimale und flexible Lagerhaltung, die schnelle Zugriffszeiten ermöglicht. Über die Mindest- und Höchstmengen gibt ein Express-Service-Katalog Auskunft, der auch die Telefax-, Telex- und TTX-Nummern enthält. Die telefonische Bestellung ist unter der Telefon-Nummer (0 89) 35 00-23 71 möglich.

Systemhauspartner

Die AEG Olympia und die Robotron business systems GmbH, hervorgegangen aus einem Zweig der Robotron Anlagenbau GmbH, präsentierten sich jetzt als Systemhauspartner. Ziel dieser Partnerschaft ist es, ein komplexes Leistungsspektrum in den Bereichen Büro- und Computertechnik anzubieten. Durch die Verknüpfung des Hardwareangebotes mit einem breiten Spektrum an Softwareerzeugnissen soll den spezifischen Anwenderbedürfnissen optimal entsprochen werden. Als wichtiger Beitrag von Rbs gehören Schulungen, zum Beispiel am Desktop-Publishing-System, sowie umfassender Service für alle AEG-Erzeugnisse zum Systemhausangebot.

Fragebogen für Netzwerk-Anwender

Aus der Erfahrung der telefonischen Beratung, die als ein Element für den Erfolg des PC-Anbieters Dell Computer anzusehen ist, hat die Firma jetzt einen Fragebogen für Netzwerk-Interessenten entwickelt. Der Fragebogen ist als Basis für eine gezielte Kundenberatung gedacht und richtet sich sowohl an solche Anwender, die erstmals den Einsatz eines PC-Netzwerkes erwägen, als auch an jene, die bereits Netzwerke unterschiedlicher Art betreiben. Die häufigsten Fragen und Unsicherheiten potentieller Netzwerk-Kunden wurden systematisiert und in logischer Abfolge erfaßt.

Den Fragebogen erhält auf Wunsch jeder Anwender, der den Langener Direktanbieter auf Netzwerkfragen anspricht. Auf der Basis der gegebenen Antworten ist es in der Regel möglich, entweder einen eindeutigen Lösungsvorschlag bis hin zum konkreten Angebot zu machen oder, in komplizierten Fällen, gangbare Alternativen mit Vor- und Nachteilen zu nennen. Im Anhang des Netzwerk-Fragebogens führt Dell einige häufig geäußerte Anwenderfragen auf und gibt Antworten, die gleichzeitig ein Grundwissen zum Thema Netzwerke vermitteln.

Leipziger CADdy Tage '90

Zur offiziellen Markteinführung der neuen CADdy Version veranstaltete ZIEGLER-Instruments im August 1990 die „Leipziger CADdy Tage '90“. CADdy-Anwendern und CAD-Interessierten in der DDR wollte das Mönchengladbacher Softwarehaus mit der Sonderveranstaltung die Gelegenheit geben, sich selbst ein Bild über die Neu- und Weiterentwicklungen der CAD-Software für Personal Computer zu machen. Gleichzeitig präsentierten namhafte Hardware-Hersteller ihre Spitzenprodukte (Grafiksysteme, Plotter, Personal Computer u. ä.). Der Messe vorgeschaltet waren spezielle Informationstage für die Vertriebspartner in der ehemaligen DDR.

Es wurde bekanntgegeben, daß das Unternehmen insgesamt 300 Programmpakete der CAD-Einsteiger-Software CADdy Junior den öffentlichen Bildungsstätten in Form einer Schenkung zukommen lassen werde. Der Gesamtwert der Software beläuft sich auf knapp 240 000 DM.

Die Aktivitäten von ZIEGLER-Instruments in der ehemaligen DDR beschränken sich nicht auf die vertriebliche Seite; auch in Sachen Programmentwicklung wird mit Partnern in der Ex-DDR zusammengearbeitet. Bereits im Sommer vergangenen Jahres wurde ein Kooperationsvertrag mit dem Forschungszentrum des Werkzeugmaschinenbaus in Chemnitz abgeschlossen. Ein Zusatzmodul zu CADdy Maschinenbau ist bereits entwickelt und Bestandteil der neuen Version. Weitere solcher Entwicklungsprojekte sind in Arbeit.

Kopierer-Tausch

Besitzern von Secop-Kopierern unterbreitet die Rank Xerox GmbH ein Angebot: Bei Abgabe des Secop-Gerätes und gegen eine verbrauchsabhängige Miete erhält der Kunde einen werksüberholten Xerox-Kopierer. Damit verbunden ist die Garantie auf Wartung des Gerätes und die Lieferfähigkeit von Ersatzteilen für mindestens drei Jahre. Die Auslieferung der Kopierer erfolgt vom Werk in Berlin-Marzahn aus. Interessenten können sich an die Rank Xerox GmbH in Ostberlin wenden (Tel. 22 52 22 25).

Nebenbei bemerkt: Die Rank-Xerox-Presseinformation beginnt mit den Sätzen: „Besitzer von SEKOP-Kopierern dürften bald erhebliche Probleme bekommen. Der Grund: Durch die Auflösung des Lieferanten VEB Robotron sind Wartung und Ersatzteilbeschaffung nicht mehr gesichert.“ Dazu Herr Füllgraf, Ge-

schäftsführer der Secop-Vertriebsfirma (Tel. Berlin 4 40 33 27): „Das ist schlicht eine Lüge! 1. Der „Lieferant" wurde nicht aufgelöst, sondern umgebildet. Der damalige VEB Robotron Vertrieb (RVB) firmierte in die Computer Vertriebsunion Berlin um. Zum 1.9. 1990 wurde als Tochtergesellschaft die CopyCom GmbH i. G. Berlin errichtet. 2. Wir, die Mitarbeiter der neu gegründeten Firma CopyCom GmbH i. G., können nachdrücklich versichern, daß der Service für Secop-Kopiergeräte sowie die damit verbundene Bereitstellung von erforderlichem Zubehör umfassend abgesichert ist. Die CopyCom GmbH garantiert allen Secop-Nutzern für jetzt und die Zukunft einen überzeugenden Kundendienst."
Selbst dann, wenn Rank Xerox handfeste Gründe hätte, daran zu zweifeln, fragt man sich, ob es die Firma nicht vielleicht nötig hat, ihre Umtauschaktion mit geschäftsschädigenden Verlautbarungen über Wettbewerber zu koppeln. *MP*

Neuer Termin für die INFOBASE '91

Die siebte INFOBASE, Internationale Fachmesse für Informationsmanagement in Frankfurt/Main, hat einen neuen Termin: 23. bis 25. April 1991. Auch inhaltlich wurde eine Änderung vollzogen: Die bisher benutzte Unterzeile *Internationale Fachmesse für elektronische Informationsprodukte* wurde ersetzt durch den neuen Titel *Internationale Fachmesse für Informationsmanagement*. Diese Umbenennung verspricht eine sinnvolle Erweiterung der Angebotspalette. Unter dem Begriff Informationsmanagement werden in Zukunft zusätzlich zum bisherigen Angebot auch Hard- und Softwarelösungen bzw. weitere periphere Produktlinien ins Ausstellerprogramm mitaufgenommen. Mit dieser Erweiterung können noch gezielter sowohl Entscheidungsträger aus Industrie und Wirtschaft als auch die zunehmend wichtig werdende Gruppe der Informationsbroker bzw. -berater angesprochen werden.

„Zilog" bleibt jetzt „Zilog"

Die Erfurter ERMIC GmbH, vormals VEB Mikroelektronik, ergänzt ihr Sortiment an Mikroprozessor- und Peripherieschaltkreisen, deren Architekturen zum großen Teil Zilog-kompatibel sind, um originale Zilogprodukte. Grundlage dafür ist ein Distributionsvertrag, der ERMIC berechtigt, als autorisierter Zilog-Distributor für Deutschland und Osteuropa aufzutreten.
Insgesamt beinhaltet die Unternehmensstrategie von ERMIC neben dem Weiterführen von Teilen der heutigen Bauelementepalette auch das Nutzen von Silicon-Foundry-Geschäften, die Entwicklung und Vermarktung eigener neuer Produkte sowie ein weites Feld von Dienstleistungen. Diese künftigen Geschäfte will man teils mit eigener Kraft, vor allem aber mit westlichen Kooperationspartnern angehen.

Ost-West-Kommunikation

Mitte September stellte sich in einer Präsentation die junge Firma Data TeleMark in Ostberlin der Öffentlichkeit vor. Die Entwicklungs- und Vertriebsgesellschaft für Telekommunikationslösungen will mit ihren Produkten und Leistungen insbesondere sich neu niederlassenden Firmen und Einrichtungen bei der Bereitstellung von Ost-West-Datenverbindungen helfen. Mit Hilfe von Kooperations- und Vertriebsabkommen mit westlichen Partnern kann somit auch der kommunikationstechnische Rückstand in den östlichen deutschen Ländern gemindert werden. Ein erster Erfolg sind beispielsweise deutsch-deutsche kommerzielle Datenverbindungen via Satellit im Zusammenwirken mit den Postverwaltungen.

PizazzPlus-Update

Für Anwender von PizazzPlus-Kopien, die ihren Wohnsitz in der Ex-DDR nachweisen, bietet die Peter Rosenthal GmbH bis zum 31.12. 1990 die Möglichkeit der Legalisierung. Zum Sonderlizenzpreis von 199,50 DM (incl. MwSt) erhält der Nutzer eine registrierte Lizenz und ein Update der neuesten PizazzPlus-Version.

NC-Standard-Schnittstelle

Die kürzlich gegründete Fördergemeinschaft SERCOS interface e. V. (FGS) hat sich zur Aufgabe gesetzt, die von einem gemeinsamen ZVEI/VDW-Arbeitskreis (einem gemeinsamen Gremium des Zentralverbandes Elektrotechnik- und Elektronikindustrie (ZVEI) sowie des Verbandes Deutscher Werkzeugmaschinenfabriken (VDW)) erarbeitete Spezifikation einer offenen digitalen Schnittstelle zur Kommunikation zwischen numerischen Steuerungen und Antrieben unter der Bezeichnung SERCOS interface zu vermarkten, das heißt, allen interessierten Kreisen, zum Beispiel Maschinenherstellern und Endbetreibern, zugänglich zu machen. Ein weiteres Ziel der Fördergemeinschaft ist es, SERCOS interface als internationalen Standard einzuführen.
Erschienen ist inzwischen auch die mehrere hundert Seiten umfassende Interfacedokumentation zum Preis von 1000,– DM (plus MwSt.).
Nähere Informationen über die Fördergemeinschaft SERCOS interface e. V. sowie die Dokumentation erhalten Sie bei der Geschäftsstelle der FGS, Stresemannallee 19, W-6000 Frankfurt am Main 70.

Umstellen der Gehaltsrechnung

Für Anwender und interessierte Unternehmen gibt die Firma Soft-Research den Informationsdienst „DDR-LOHNAKTUELL" heraus. Er behandelt aktuelle Probleme des Umstellens der Lohn- und Gehaltsabrechnung. Der Informationsdienst ist kostenlos und kann formlos angefordert werden bei: Soft-Research GmbH, Leopoldstraße 56A, W-8000 München 40; Tel. (0 89) 34 98 22.

Pressearbeit – aber wie?

Das Ergebnis zahlreicher Presseinformationen ist für Redaktion wie Anbieter gleichermaßen frustrierend: Marktschreierische Übertreibungen gehen oftmals einher mit unprofessioneller Aufbereitung. Die Überarbeitung der Texte ist oftmals zu aufwendig, so daß sie häufig nicht berücksichtigt werden. Abhilfe schaffen will das Impuls-Seminarzentrum LWP in Lenggries mit einem „Presse-Workshop für DV-Unternehmen". Neben Pressetexten sollen der Aufbau einer Strategie für Öffentlichkeitsarbeit, der Kontakt mit Journalisten, die Organisation von Pressekonferenzen und andere PR-relevante Themen speziell der DV-Branche besprochen werden.
Hardwarehersteller, Software- und Systemhäuser sowie Dienstleistungsunternehmen, die Tips für ihre Pressearbeit suchen, wenden sich an das Impuls-Seminarzentrum LWP GmbH, Ludwig-Thoma-Weg 9, W-8172 Lenggries, Tel. 0 80 42/80 81.

Tandberg-Data-Peripherie am ESER

In Zusammenarbeit mit dem Datenverarbeitungszentrum (DVZ) in Leipzig entwickelte die Tandberg Data GmbH eine Lösung, um ihre IBM 3174-kompatible Peripherie an die in der Ex-DDR verbreiteten ESER-Zentraleinheiten anschließen zu können. Die Anpassung der Steuereinheit an den ESER-Rechner erfolgt dabei über Software. Dies eröffnet den Betreibern von Rechenzentren in der Ex-DDR die Möglichkeit, die Zentraleinheit zu einem späteren Zeitpunkt gegen aktuelle IBM-Modelle oder kompatible auszutauschen. Lediglich ein Wechsel der Software ist dann erforderlich.

Rund um Clipper

Die Firma Nantucket bietet jetzt einen neuen Add-on-Katalog im Taschenbuchformat an, der in weiteren Ausgaben ständig aktualisiert wird.
Neben Informationen über Tools und Utilities von Drittanbietern enthält das Nachschlagewerk eine Übersicht über das umfangreiche Angebot an ausgereiften und interessanten in Clipper erstellten Applikationen. Zusätzlich sind ein Verzeichnis über deutschsprachige Literatur und ein detaillierter Index enthalten.
In übersichtliche Kapitel gegliedert, weisen die einzelnen Einträge die Bezugsadresse, eine Kurzinformation sowie die Preisangabe für das jeweilige Produkt auf. Zu beziehen ist *Rund um Clipper* zum Preis von 19,80 DM (inkl. MwSt) bei der Firma Nantucket GmbH, Postfach 220101, W-5090 Leverkusen 22.

Technologiestudien

Im Auftrag des Bundesministeriums für Forschung und Technologie (BMFT) hat die VDI/VDE-Technologiezentrum Informationstechnik GmbH insgesamt 14 Studien zur Mikrosystemtechnik erstellen lassen. Sie vermitteln eine Übersicht über den nationalen und internationalen Entwicklungsstand sowie Trends der Mikrosystemtechniken. Innovationsgehalt und -risiken, Marktanforderungen und -prognosen, Technologieanbieter und Dienstleister – auch diese Aspekte werden in den Berichten behandelt.
In der Studie ASICs wird versucht, den aktuellen Stand der ASIC-Technik und das darin enthaltene Potential kleiner und mittlerer Unternehmen darzustellen; sie geht aber auch insbesondere auf Problempunkte ein, die ein potentieller ASIC-Entwickler oder -Anwender zu beachten hat. Sie ist so gegliedert, daß nach einer Begriffsdefinition der Stand der Technologieentwicklung bei anwenderspezifischen Schaltkreisen aufgezeigt wird. Hierbei sollen die hinter dem Stichwort „Entwicklungsumgebung" verborgenen Begriffe wie Design-, Simulations- und Testwerkzeuge, Aufgabenteilung Hersteller – Anwender und mögliche Schnittstellen dieser Arbeitsteilung beleuchtet werden.
Der Abschnitt „Verfügbarkeit" stellt eine Art Checkliste der Entscheidungs- und Auswahlkriterien für eine ASIC-Entwicklung dar. Neben den rein technischen Details spielen hier auch Fragen der Personalverfügbarkeit und erforderlichen Investitionshöhe eine Rolle. Im letzten Teil dieser Studie schließt sich eine Darstellung der Technologielandschaft in der Bundesrepublik Deutschland an, wobei besonders die Dienstleistungen im ASIC-Bereich präsentiert werden.
Weitere Studien gibt es u. a. zu den Themen
– piezokeramische Aktoren
– Mikromechanik
– intelligente Module und anwendungsorientierte Leistungshalbleiter.

Die Studien können für jeweils 60,– DM über folgende Anschrift bezogen werden: VDI/VDE-IT GmbH, Herr Beumler, Budapester Straße 40, W-1000 Berlin 30; Tel. 2 60 91 94.

Neue Computermesse in Asien

Auch Asien hat jetzt seine eigene Prestige-Messe für Informations- und Kommunikationstechnik: die **CENIT ASIA '90**, die vom 25. bis 28. September in Hongkong stattfand. Etwa 250 Aussteller aus 26 Ländern zeigten ihre neuesten Produkte auf einer Fläche von 20 000 m^2. CENIT ist die asiatische Version der weltweit größten Messe für Informations- und Kommunikationstechnik, der CeBIT, die jedes Jahr in Hannover abgehalten wird. Die CENIT wird gemeinsam von der Hannover-Messe International GmbH und der Firma Adsale Exhibition Service Ltd. organisiert. Gleichzeitig mit der Premiere dieser Messe wurden internationale Konferenzen und Symposien abgehalten. Beispielsweise die *Region Ten Conference* mit dem Thema Computer und Kommunikation, organisiert vom Institut der Elektro- und Elektronikingenieure (IEEE – Institute of Electrical and Electronics Engineers Inc.), dem mehr als vierzehn Mitgliedsländer und 23 000 Einzelmitglieder in Asien angehören.

Für Studenten billiger

TommySoftware bietet jetzt seine deutschen Softwareprodukte als kostengünstige Studentenversionen an. Das Angebot gilt auch für Schüler. Alle Studentenversionen sind um rund 50 % gegenüber den normalen Versionen gesenkt worden. Sie bieten dabei die gleiche Leistung. MegaPaint II wird als Studentenversion für 249,– DM vertrieben, MegaPaint II Professional für 399,– DM, MegaPaint II PC für 289,– DM, und die Musikprogramme SoundMachine II und SoundMerlin werden für 99,– DM bzw. 149,– DM abgegeben.

Leiterplattenentwurf am PC

Lutz Mantz, Berlin

Eine Zahl hat mich überrascht: Nach übereinstimmenden Einschätzungen mehrerer Kleinfertigungszentren und verschiedener Hersteller wurden zum Jahresanfang in der damaligen DDR noch 90 % aller Leiterplattenkonstruktionen als Handentwurf gefertigt. Die Ursachen dafür sind offensichtlich vielfältig: zum ersten die bislang ungenügende Bereitstellung grafikfähiger Rechner, sowohl in Form von PCs als auch von Workstations, zum zweiten scheinen die Ansprüche vieler Entwickler noch mit Einlagenleiterplatten niedriger Schwierigkeitsgrade zu befriedigen zu sein, zum dritten waren zu diesem Zeitpunkt die Herstellungskapazitäten für Leiterplatten höherer Schwierigkeitsgrade auch nicht gerade reichlich vorhanden, und zum letzten standen noch zu wenige Informationen über Sinn und Zweck des rechnergestützten Leiterplattenentwurfes, die Anforderungen, die an solche Systeme gestellt werden, und deren Leistungsfähigkeit zur Verfügung.

Das richtige System entscheidet

Nun wollen wir in diesem Artikel keine Patentrezepte entwickeln und nicht am Ende ermitteln: Dieses eine System erfüllt alle Ansprüche. Ein absolut ideales System gibt es nicht, und was heißt schon „absolut ideal"? Dazu sind die Anforderungen in den Unternehmen zu verschieden, so daß es sinnvoller erscheint, auf einige Grundsätze bei der Auswahl hinzuweisen und sie zu diskutieren. Da den Entwicklern ein vielfältiges Spektrum an Leiterplattensoftware vorliegt, gilt es, recht gründlich zu überlegen, ob und für welches Entwurfssystem man sich entscheiden soll.
Ausgangspunkt unserer Überlegungen soll sein, daß wir von Systemen ausgehen, die für den professionellen Einsatz gedacht sind. Ferner stehen Systeme im Vordergrund, die auf PC-Basis einsetzbar sind, was durchaus kein Qualitätskriterium darstellt, sondern für Anwender in mittleren Unternehmen und für Einsteiger schon allein aus Kostengründen eher zu empfehlen ist. Denn wenn man sich für den rechnergestützten Konstruktionsentwurf entscheidet, sollte man nicht gleich „mit Kanonen auf Spatzen schießen".

Zeichnen kostet Zeit

Den aufwendigen Weg des manuellen Entwurfes hier detailliert zu beschreiben erübrigt sich von selbst, denn der dürfte hinlänglich bekannt sein. Allein der Zeitaufwand für eine optimale Plazierung im Hinblick auf kürzeste Leiterzugverbindungen, eine geringe Anzahl von Leiterzugkreuzungen und eine Bauelementeanordnung, die den Einbaubedingungen der Leiterplatte im Gehäuse entsprechen, ist erheblich. Bekanntlich wird durch eine günstige Plazierung der Bauelemente der Aufwand für die Entflechtung (Trassierung, Routing) positiv beeinflußt – abgesehen davon, daß für funktionstüchtige Leiterplatten, besonders in den Bereichen HF-Technik und Digitaltechnik, einige Erfahrungen notwendig sind. Trotzdem ist die Leiterplattenkonstruktion selbst bei einer Zweiebenenleiterplatte mit Durchkontaktierungen (DKL) mit wenigen hundert Verbindungen eine zeitaufwendige und mühselige Angelegenheit. Und zum Schluß sind noch die Fertigungsunterlagen in der entsprechenden Präzision anzufertigen. Damit wären wir bereits beim Erstellen der vielen Unterlagen, die insgesamt für einen Entwurf benötigt werden. In kleineren Unternehmen kann in speziellen Fällen auf das eine oder andere verzichtet werden, aber insgesamt sind doch unter anderem eine Menge Zeichnungen anzufertigen, die jede für sich als Unikat angesehen werden muß.
Von den vielen spezifischen Arbeitsmitteln und Unterlagen seien hier nur die wichtigsten herausgegriffen. Als erstes wäre der Stromlaufplan zu nennen, der auf dem Zeichentisch aus der Handskizze entstehen soll. Dazu gehören die Bauteilliste für die Materialbeschaffung und den Konstrukteur, der dazu noch eine Verbindungsliste benötigt, die bei größeren Konstruktionen schon recht umfangreich werden kann. Aus der eigentlichen Konstruktion entstehen die Leiterbildzeichnung, Bohrpläne (Bearbeitungszeichnung), die Bestückungszeichnung, die Lötstopmaske, die Siebdruckmaske usw., womöglich Steuerdatenträger für Lichtzeichenmaschine, Plotter und Bohrmaschine – und das für alle Ebenen. An Steuerdatenträger für Tester und Bestückungsautomaten sei noch gar nicht gedacht. Bei Einebenenleiterplatten mag das alles noch im Rahmen des Möglichen bleiben, aber bereits bei Zweiebenenleiterplatten oder sogar Mehrebenenleiterplatten mit 4 und mehr Lagen versagen die manuellen Methoden schon allein wegen der Unübersichtlichkeit und der mangelnden Präzision. Schon bei einer Verschiebung der zweiten Lage um einen Rasterschritt im Konstruktionsraster (z. B. 1/20 Zoll) sind die Glasoriginale bzw. Folienkopien nutzlos, weil in der Fertigung nichts mehr zusammenpaßt.

Wofür brauche ich den PC?

Ein simples Zeichensystem mit Bildschirmunterstützung dürfte wohl keine Abhilfe schaffen. Hier sollten wir bedenken, daß ein PC ja nicht nur „Rechenleistungen" vollbringt, die wir im Entwurfsprozeß durchaus noch schätzen lernen werden, sondern daß auch die Speicherkapazitäten von Festplatte und Diskette intensiv zur Ablage genutzt werden können. Wie ein PC aussehen sollte, der für den Leiterplattenentwurf geeignet ist, werde ich später erläutern.
Was aber ist denn so aufhebenswert? Zunächst das, was von einem Entwurf in Papierform ohnehin aufgehoben wird: der Stromlaufplan, die Darstellung der Leiterzugbilder für alle Ebenen (Layout), die Bestückungszeichnung usw., die somit für Sie jederzeit reproduzierbar sind.
Aber es lohnt sich, noch mehr aufzuheben, nämlich Informationen, die bei jeder Konstruktion aufs neue benötigt werden: die Schaltplansymbole, die geometrischen Bauelementedaten, kurz: die Bauelementebeschreibungsdaten. Und wir werden sehen, daß da noch eine ganze Menge anderer Informationen hinzukommen kann.

Der Bauelementekatalog

Die Aufbewahrung der Informationen zu den Bauelementen erfolgt bei rechnergestützten Entwurfssystemen in Bauelementekatalogen, die unabhängig von den Entwurfsdaten gehalten und verwaltet werden. Die Struktur, die Organisation, die Speicherkapazitäten und die Verwaltungsmechanismen der Bauelementekataloge sind in den einzelnen Systemen unterschiedlich. Wichtig aber ist, daß einmal erfaßte Informationen über die Bauelemente für alle späteren Konstruktionen wieder zur Verfügung stehen. Das bedeutet, die Stromlaufplansymbole, die elektrischen und geometrischen Beschreibungen der analogen und digitalen Bauelemente, die in einem Unternehmen benötigt und verwendet werden, liegen in einer für das Entwurfssystem – und natürlich für den Konstrukteur – auswertbaren Form vor. Als zweckmäßig hat sich in der Praxis eine Struktur erwiesen, die etwa wie in Bild 1 dargestellt aussieht.

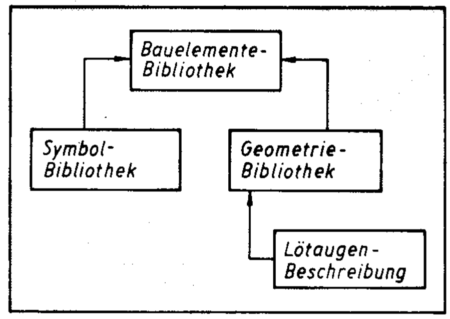


Bild 1 Prinzipieller Aufbau eines Bauelementekataloges

Was gehört in die Kataloge?

Die Symbolbibliothek enthält die grafische Beschreibung der Stromlaufplansymbole. Ein gutes Entwurfssystem läßt Stromlaufplansymbole als Bauelementeäquivalente (siehe Bild 2) und logische Funktionsbeschreibungen, die den Gattern der digitalen Bauelemente entsprechen, zu. Dazu kommen alle Textinformationen, die konstant bleiben, sowie die Positionen für Textinformationen, die erst im Stromlaufplan hinzugefügt werden (z. B. Schaltteilkurzzeichen). In einer Geometriebibliothek sollten die grafischen Elemente Plazierungsumriß, Bestückungszeichnungssymbol, Lage der Bauelementeanschlüsse (Pins) sowie Textinformationen und zum Bauelement gehörende Kupferflächen enthalten sein.
Unterschiedlich wird die Beschreibung der Lötaugen an den Pins gehandhabt. Bei eini-

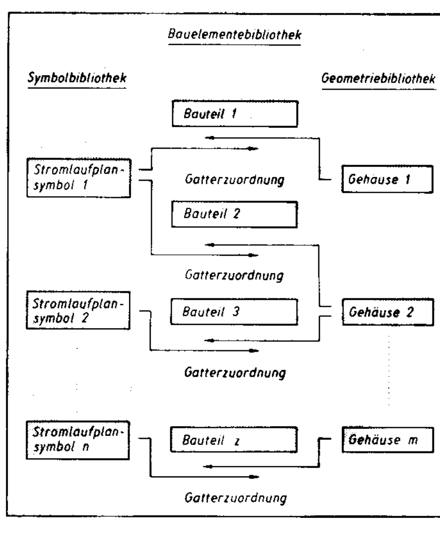


Bild 2 Beispiel für ein Stromlaufplansymbol

gen Systemen sind Lötaugenform und -größe bereits mit der Geometriebeschreibung des Bauelementes gekoppelt. Das erweist sich aber in Konstruktionsabteilungen, wo Leiterplatten unterschiedlicher Schwierigkeitsgrade entwickelt werden, als Einschränkung. Günstiger ist die Arbeit mit Systemen, bei denen die konkrete Lötaugenbeschreibung erst im Entwurf gemacht wird, und das nach Möglichkeit für Außen-, Innen- und Spannungs-/Masseversorgungslagen getrennt. Darüber hinaus sollten auch unabhängige Lötaugendefinitionen zur technologieabhängigen Festlegung der Größe von Durchkontaktierungen und anderen, technologisch notwendigen Bohrungen mit und ohne Metallisierung möglich sein.

Die Bauelementebibliothek beinhaltet die Beschreibung des konkreten Bauelementespektrums. Hier werden die logischen Informationen (Stromlaufplansymbol) mit den physischen Informationen (Bauelementegeometrie) zusammengeführt, das heißt, pro Bauelementetyp erfolgt eine Eintragung. Dazu gehört die Zuordnung der Anschlüsse am Stromlaufplansymbol zu den Pins des Bauelementes einschließlich der Beschreibung von Gatterstruktur und -anzahl, möglichst die Funktionsbeschreibung der Pins (Input, Output, bidirektional, unbelegt, Spannung, Masse usw.). Einige Systeme ordnen einzelnen Pins konkrete Signalnamen zu, wie bei Spannung (z. B. +12 V, +5 V) und Masse (z. B. GND), da ja die Anschlüsse nicht immer am Stromlaufplansymbol ersichtlich sein müssen, zum anderen aber bei den einzelnen (digitalen) Bauelementen die Spannungs- und Massepins festliegen und auch die Betriebsspannung der (digitalen) Bauelemente im allgemeinen konstant ist.

Wie wirken nun die einzelnen Katalogeinträge zusammen? Im allgemeinen wird jedes Stromlaufplansymbl, jede Geometrie und jede Lötaugenbeschreibung nur einmal erfaßt und mit einem Namen in den speziellen Katalogteilen abgelegt. Bei der Beschreibung der Bauelemente wird dann auf diese abgelegten Elemente zurückgegriffen. Das kann ungefähr so aussehen wie in Bild 3.

Erst wenn die Kataloge stimmen, kann's losgehen

Mit der Verflechtung der Katalogteile und ihrer Inhalte entsteht natürlich auch ein Risiko:

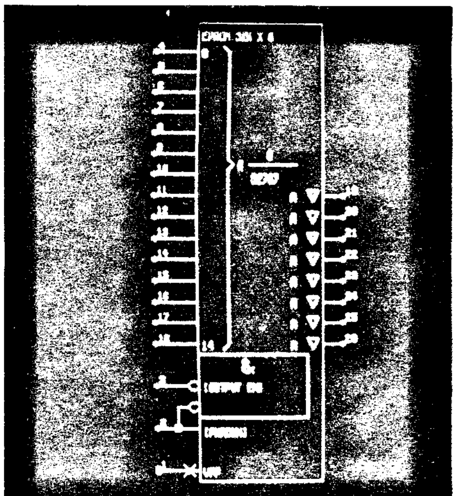


Bild 3 Einmal erfaßte Symbole und Geometrien können beliebig den Bauteilbeschreibungen zugeordnet werden

Wenn auch nur eine Bauelementegeometrie falsch im Katalog beschrieben wird, entstehen in allen Konstruktionen, die dieses Bauelement verwenden, Fehler. Aber: Was einmal richtig in der Bibliothek drin ist, ist eben auch beim hundertsten oder tausendsten Verwenden richtig.

Beim Erwerb eines Entwurfssystems sollten Sie darauf achten, daß ein Katalogverwaltungssystem mit dazu gehört, mit dem Stromlaufplansymbole, Geometrien, Bauelementebeschreibungen und eventuell Lötaugenformen zum Katalog hinzugefügt, darin modifiziert und wieder gelöscht werden können. Besteht nämlich diese Möglichkeit nicht, sind Sie immer von der Katalogaktualisierung des Anbieters abhängig, was Geld und Zeit kosten kann. Wesentlich günstiger ist es natürlich, wenn man wenigstens den Grundstock eines Katalogsystems, das zum Entwurfssystem paßt, käuflich erwerben kann. Im allgemeinen sind solche Kataloge nicht gerade billig, aber die Kosten haben sich schnell wieder amortisiert. Denn ohne einen Katalog ist ein Entwurfssystem nicht einsetzbar, und die Ausgaben für das Entwurfssystem bringen erst einen Nutzen, wenn mit dem System gearbeitet wird. In der Zeit der Katalogerfassung ist aber keine produktive Arbeit mit dem System möglich, selbst eine ernstzunehmende Einarbeitung und Qualifizierung am System ist kaum vorstellbar. Deshalb würde ich den Erwerb eines Kataloges stets empfehlen, und wenn er nur dazu vorliegt, herauszubekommen: Wie wird's denn überhaupt gemacht? Genauer sollte man aber schon prüfen, ob man sich in jedem Fall auf einen Katalogwartungsvertrag einläßt. Hier müßte von Fall zu Fall entschieden werden, ob sich das rentiert. Das hängt in erster Linie vom Aufgabenspektrum und von der Innovationsfreudigkeit des Unternehmens ab.

Der Stromlaufplan

Nun wissen wir, daß so ein Katalog eine Sache ist, in die eine Menge Aufwand hineingesteckt werden muß. Wann zahlt sich denn so etwas überhaupt aus? Das erste Mal bei der Erfassung des Stromlaufplanes. Voraussetzung dafür ist natürlich nach wie vor der Schaltungsentwurf als Handskizze. Alle professionellen Systeme besitzen einen grafischen Editor für die Erfassung des Stromlaufplanes.

Zweckmäßig ist es, auch standardgerechte Formblätter (Zeichenblätter) als Zeichnungselement einmal zu erfassen und in der Bibliothek abzulegen. Damit ist der erste Schritt der Aufruf des Formulars. Es wird mit Hilfe des Cursors auf dem grafischen Bildschirm an der gewünschten Stelle abgelegt. Jetzt können die Stromlaufplansymbole für die logischen Funktionen und die Bauelemente mit ihren Symbolnamen aufgerufen und an der gewünschten Stelle auf dem Formular abgesetzt werden. Die meisten Systeme haben eine Wiederholfunktion, so daß ein mehrfach benötigtes Symbol nur einmal aufgerufen werden muß und mehrfach an den jeweiligen Stellen abgesetzt wird. Meist ist auch eine Kopierfunktion für Gruppen von Symbolen vorhanden, so daß bereits erfaßte funktionelle Gruppen auf dem Bildschirm mehrfach verwendet werden können. Bei Symbolen, die komplette Bauelemente repräsentieren, brauchen jetzt nur noch die Schaltteilkurzzeichen an den Symbolen untergebracht zu werden.

Etwas komplizierter ist das Ganze bei logischen Symbolen, bei denen ja keine eindeutige Zuordnung zu einem Bauelementetyp vorliegt. Hier muß neben dem Schaltteilkurzzeichen auch noch der Bauteiltyp angegeben werden. Viele Systeme bieten auf der Grundlage der Kataloginformationen alle in Frage kommenden Bauteiltypen an. Bei anderen ist man gezwungen, mit Hilfe der Informationslisten über die Kataloginhalte den gewünschten Bauteiltyp herauszusuchen. Da aber für einen Entwurf bereits meist bei der Vorbereitung der Konstruktion feststeht, welche Bauelemente verwendet werden sollen, sind für diesen Arbeitsschritt die notwendigen Informationen ohnehin schon vorhanden.

Bevor man sich für ein System entscheidet, sollte man besonders nach der Systemintelligenz für die Bearbeitung logischer Funktionen fragen: Gute Systeme zählen nämlich die zu einem Bauelement gehörenden Gatter mit und machen darauf aufmerksam, daß ein neues Bauelement (und ein neues Schaltteilkurzzeichen) benutzt werden muß, wenn die Gatteranzahl des vorhergehenden Bauelementes erschöpft ist. Sie ordnen auch die physische Pinnummer des Bauelementes den Stromlaufplansymbol-Anschlüssen zu. Dieser Automatismus wird nicht unbedingt immer gewünscht sein. Deshalb gehört auch eine manuelle Pinzuordnungsmöglichkeit dazu, was ganz besonders bei Steckverbindern wichtig ist, weil ja hier die Pinbelegung durch das technische Gesamtkonzept bestimmt wird.

Bei der Handhabung von analogen Bauelementen sieht die ganze Sache allerdings etwas anders aus. Hier unterscheiden sich auch die Katalogkonzepte erheblich. Einige Systeme verlangen, daß alle verwendeten Bauelemente komplett beschrieben werden. Das erfordert natürlich einen enormen Katalogpflegeaufwand, wenn man nur an alle Ausführungen einer einzigen Widerstandsbaureihe denkt. Hier scheint die Lösung in Systemen, wie Cadstar von Racal-Redac günstiger zu sein, wo in der Bauelementebibliothek nur integrierte Bauelemente oder Bauelemente mit einer funktionell definierten Pinzuordnung erfaßt sind.

Widerstände, Kondensatoren und andere Bauelemente werden nur als Symbole dargestellt, und ihre vielfältigen Geometrien werden in den jeweiligen Bibliotheken gehalten. Bei der Stromlaufplanerzeugung wird dann nur das Symbol *Widerstand* aufgerufen und statt einer Bauelementebeschreibung nur die gewünschte Bauelementegeometrie zugeordnet. Der Widerstandswert, die Belastbarkeit und der Toleranzwert werden als Textinformationen verbal hinzugefügt, denn diese Werte spielen bei der Leiterplattenkonstruktion im allgemeinen keine Rolle, von Interesse sind hier nur die geometrischen Informationen für die Plazierung. Nachteilig ist allerdings, daß man sich bereits im Stromlaufplan entscheiden muß, ob beispielsweise ein konventioneller Widerstand liegend oder stehend verwendet werden soll. Bei Cadstar und anderen professionellen Systemen besteht jedoch die Möglichkeit, unter anderem Bauelemente und Geometrien noch während des Leiterplattenentwurfes auszutauschen, so daß hier recht große Freiheitsgrade bestehen und die oben genannte Einschränkung wieder ausgeglichen wird.

Wenn die Bauelementezuordnung abgeschlossen ist, werden die Verbindungen zwischen den Symbolen hergestellt. Hier gilt es natürlich, mit großer Exaktheit vorzugehen,

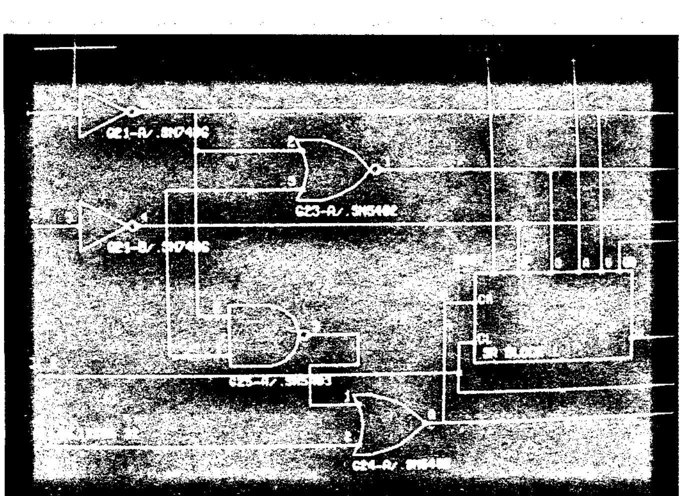
denn ein Fehler, der sich hier einschleicht, ist nur noch schwer zu beheben, wenn man ihn während des Konstruktionsprozesses überhaupt noch bemerkt. Einige Systeme verlangen für jede Verbindung einen Netz- oder einen Signalnamen, bei anderen sind nur ausgewählte Verbindungen zu benennen, zwingend aber bei der Verwendung von Busstrukturen. Busverbindungen werden in den Systemen jedoch nur als Zeichenhilfen benutzt, es leiten sich daraus noch keine Strukturen auf der künftigen Leiterplatte ab.
Nun wird es in den seltensten Fällen gelingen, Stromlaufpläne für komplexe Schaltungen auf nur einem Zeichnungsblatt unterzubringen. Die derzeit gängigen Systeme sind alle so angelegt, daß mehrere Zeichnungsblätter zu einer Konstruktion zusammengefaßt werden können (horizontale Verknüpfung). Vielfach besteht auch die Möglichkeit, Stromlaufpläne in Hierarchien aufzubauen, womit Blockschaltbilder erzeugt werden können. Der Konstrukteur kann dann festlegen, welcher bereits vorhandene und auf Diskette abgespeicherte Stromlaufplan für diesen Schaltungsblock vom System in die Gesamtschaltung integriert werden soll (siehe Bilder 4 und 5).
Über solche Strukturierungsmechanismen ist es aber auch möglich, bestimmte, immer wiederkehrende Schaltungsteile einmal zu erfassen und sie bei Bedarf in komplexe Schaltungen einzubinden (ein ganz simples Beispiel ist die Stromversorgung bei einer bestimmten Gerätegeneration).

Jetzt zahlt sich die Vorbereitung aus

Aus dieser Vorgehensweise wird indirekt auch klar, daß einmal erfaßte Stromlaufpläne auf Festplatte oder auf Diskette abgespeichert werden können und damit jederzeit wieder zur Verfügung stehen. Über diesen Weg ist es natürlich auch relativ einfach möglich, vorhandene Schaltungen wieder aufzurufen, sie zu modifizieren und unter einem neuen Namen innerhalb eines neuen Projektes abzuspeichern, ohne den ursprünglichen Entwurf dadurch zu verändern. Daraus wird auch deutlich, daß der eigentliche Produktivitätsgewinn mit solchen Systemen nicht auf Anhieb, sondern durch kontinuierliche Arbeit über einen längeren Zeitraum eintritt. Dann erst entsteht auch die notwendige Routine, die für die Handhabung eines solchen Systems erforderlich ist, und auch die Kenntnis über alle „Ecken und Winkel" des Systems, die es dem Konstrukteur erlaubt, das System in seiner Leistungsfähigkeit „auszureizen".
Aber kommen wir zurück zum Entwurfsprozeß. Die auf dem Bildschirm sichtbare und auf Diskette abgespeicherte Schaltung ist zwar recht nett anzuschauen, aber nützt auch noch relativ wenig. Zwei Dinge sind jetzt erforderlich: Die Zeichnung des Stromlaufplanes und maschinenlesbare Informationen für die Leiterplattenkonstruktion. Die Zeichnung ist relativ einfach über einen Plotter zu erzeugen, die dazu notwendigen Hilfsmittel sind eigentlich in allen Systemen enthalten. Allerdings sollte man sich vor dem Erwerb sehr genau über die Hardwarebedingungen und die unterstützten Plottertypen informieren, damit man an dieser Stelle nicht eine böse Überraschung erlebt. Wir gehen auf diese Dinge später noch einmal etwas detaillierter ein.
Wesentlich weniger Einfluß hat man aber auf die Übergabe der Stromlaufplandaten zum Leiterplattenentwurf. Deshalb sollte man hier stets darauf achten, daß Stromlaufplanentwurf und Leiterplattenentwurf als gemeinsames, homogenes System vom gleichen Hersteller angeboten werden. Hier sind ordentliche Übergabebedingungen garantiert, die auch die Kataloginformationen gründlich auswerten. Im allgemeinen werden beim Übergang vom Stromlaufplan zum Leiterplattenentwurf vom System automatisch eine Bauteilliste und eine Verbindungsliste erzeugt. Diese Daten werden durch das System mit den geometrischen Bauelementebeschreibungsdaten ergänzt. An dieser Stelle sehen wir, wie wichtig eine korrekte Katalogpflege ist und welchen Stellenwert der Katalog in einem solchen System besitzt. Um es noch einmal deutlich zu sagen: Ausgangspunkt der Bauteil- und der Verbindungsliste ist der grafisch erfaßte Stromlaufplan. Die manuelle Erzeugung solcher Listen entfällt. Damit wird die Fehlerquote in der Verbindungsliste auf Null gesetzt – vorausgesetzt natürlich, daß der Stromlaufplan ordentlich erfaßt worden ist.
Bevor wir zum eigentlichen Leiterplattenentwurf kommen, sei hier noch eine Bemerkung gestattet. In vielen Gesprächen wurde von Konstrukteuren und Schaltungsentwicklern der Wunsch nach einem Simulationssystem geäußert. Ich möchte in diesem Artikel die Problematik Simulation nicht weiter diskutieren, denn sie ist selbst eine Artikelserie wert. An dieser Stelle möchte ich nur allzugroße Erwartungen in Simulationssysteme auf PC-Basis dämpfen, denn die Simulation des Signalspiels eines einzigen Bauelementes – geschweige denn einer ganzen Baugruppe – erfordert enorme Rechenleistungen. Das hängt damit zusammen, daß eine große Menge an bauteilbeschreibenden Daten für die Simulation notwendig ist. Einige davon lassen sich nicht als arithmetische Größen, sondern nur als Funktionen (im mathematischen Sinn) oder komplexe Funktionsbeschreibungen darstellen. Die selbständige Pflege dieser Daten ist nur in wenigen Fällen ökonomisch vertretbar, wenn überhaupt durchführbar. Die Daten müssen von den Simulationssystemen ausgewertet werden, was eine zeitaufwendige Angelegenheit ist. Andererseits müssen diese Daten ja erst einmal erfaßt werden, um sie für die Simulation zur Verfügung zu haben. Hier muß jedes Unternehmen selbst entscheiden, ob dieser Aufwand in einer vertretbaren Relation zum Nutzen steht. Deshalb kommt hier auch in erster Linie eine Simulation der logischen Funktionen in Frage, die noch in vertretbaren Aufwandsgrößen zu bewältigen ist.

Wir beginnen das Layout

Ebenso wie bei der Erfassung des Stromlaufplanes stehen bei der eigentlichen Leiterplattenkonstruktion die grafischen Methoden im Vordergrund. Jedoch im Gegensatz zum Stromlaufplan werden bei der Leiterplattenkonstruktion nicht mehr die logischen Schaltplansymbole, sondern die geometrischen Informationen der Bauelemente als grafische Objekte verwendet. Weitere wichtige Objekte sind die aus den logischen Verbindungen entstehenden physischen Kupfertrassen (Routes), Kupferflächen (beispielsweise Schirmflächen), der Leiterplattenumriß, bei einigen Systemen die Lötaugen, aber auch Texte, die auf die Platine geätzt werden sollen. Am Beginn der Konstruktionsaufgaben steht in jedem Fall die Festlegung der Konstruktionsparameter. Dazu gehören (ohne Vollständigkeit): die Trassenbreiten der Signal- und Stromversorgungsverbindungen, die minimalen Abstände zwischen Pin und Pin, Pin und Trasse, Trasse und Trasse sowie zu Kupferflächen, das Plazierungsraster, das Konstruktionsraster, Form und Größe der Durchkontaktierungen, die Anzahl der Leiterplattenebenen, Vorzugsrichtungen auf den einzelnen Ebenen usw. Gerade bei der Flexibilität und der Handhabung der Einstellung der Konstruktionsparameter unterscheiden sich die Systeme erheblich. Hier sollten sich die künftigen Anwender vor dem Erwerb eines Systems sehr gründlich informieren. Worauf ist hierbei besonders zu achten?



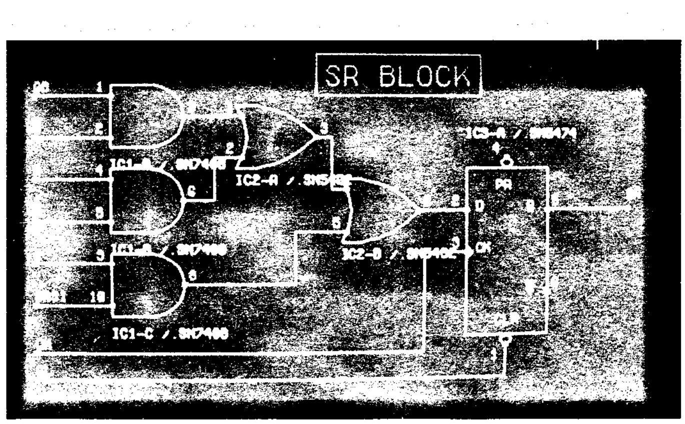


Bild 5 Inhalt des Schaltungsblocks von Bild 4

◀ *Bild 4 Stromlaufplan mit Schaltungsblock*

Einige Systeme bieten nur wenige feste Trassierungsraster an (1/20", 1/40", 1/80"), die nicht verändert werden können, sondern zwischen denen nur ausgewählt werden kann. Auch die maximal und minimal möglichen Abmaße für Trassen und Lötaugen sollten genau geprüft werden. Bei einigen Systemen besteht nach Beginn der Konstruktionsarbeit keine Möglichkeit mehr, die Parameter zu verändern. Günstig dagegen sind Systeme, die mehrere Signaltrassenbreiten (die auf den Lagen auch unterschiedlich sein können) zulassen, bei denen der Wechsel des Arbeitsrasters bis zu einer Feinheit von 1/1000 Zoll möglich ist sowie Form und Größe der Lötaugen und Durchkontaktierungen frei gewählt werden können. Zunehmend spielt auch die Verwendung mehrerer, unterschiedlicher Durchkontaktierungen (Blindvias, verdeckte Durchkontaktierungen) eine Rolle. Achten Sie auch bei der Angabe der maximal möglichen Anzahl von „Leiterplatten-Ebenen" darauf, ob es sich dabei tatsächlich um Ebenen handelt, die der Beschreibung elektrischer Objekte dienen.
Wichtig bei der Parametrisierung ist auch, wie groß die Unterstützung des Systems in Richtung Postprozeß ist. Das klingt in diesem Zusammenhang zunächst wenig einleuchtend. Doch was nützen Ihnen die interessantesten Parametereinstellungen, wenn es Ihnen beispielsweise nicht gelingt, bestimmte Informationen auf die Lichtzeichentechnik zu übertragen, oder wenn Sie vom System keine Warnung erhalten, daß die Blende der Lichtzeichenmaschine für eine bestimmte Trassenbreite gar nicht vorhanden ist? Ferner sollte eine Möglichkeit bestehen, komplette Parametersätze für unterschiedliche Schwierigkeitsgrade oder Hersteller auf der Diskette abzulegen, um diese vor Konstruktionsbeginn in den Datenbestand der Konstruktion einlesen zu können.
Eine Bemerkung möchte ich noch zum Arbeitsraster der Systeme machen. Bekanntlich gibt es Probleme bei der sogenannten Mischbestückung: Auf einer Platine im Zollraster sollen metrische Bauelemente oder umgekehrt verwendet werden. Deshalb ist es günstig, sich für ein System zu entscheiden, das ein Arbeitsraster bis zu 1/1000 Zoll ermöglicht, denn dann können zum Beispiel die metrischen Konstruktionselemente hinreichend genau in Zollmaßen dargestellt werden.

Die Leiterplattenkontur muß her

Nehmen wir an, die Konstruktionsbedingungen sind eingestellt. Was wir nun brauchen, ist die Leiterplattenkontur. Sie wird entweder auf dem Bildschirm gezeichnet, oder die in einem Unternehmen üblichen Leiterplattenkonturen sind auf den Datenträgern der Konstruktionsrechner abgelegt und stehen somit bereits zur Verfügung. Inzwischen sind bei fast allen Systemen freie Darstellungsmöglichkeiten für den Leiterplattenumriß möglich, also auch abgerundete Ecken oder sogar kreisförmige Umrisse. Aber wie nun weiter? Die aus dem Stromlaufplan erzeugten Bauelemente- und Verbindungslisten werden jetzt vom System auf der Grundlage der Bauelementeinformationen aus dem Katalog verarbeitet. Im Ergebnis dessen erscheinen unsere Bauelemente in grafischer Form auf dem Bildschirm. Die logischen Verbindungen aus dem Stromlaufplan werden als Luftlinien zwischen den Bauteilpins dargestellt. Bei einigen Systemen wird die grafische Darstellung der Bauelemente erst über die Angabe des Schaltteilkurzzeichens aktiviert. Unter Umständen ist es jetzt noch notwendig, konstruktionsbedingte Kupferflächen (Schirmflächen oder ähnliches) sowie Flächen, die für Bauelemente oder Trassen gesperrt sein sollen, festzulegen.

Starke Hilfe für das Plazieren

Nun kann das Plazieren der Bauelemente beginnen. Dafür stehen manuelle und vielfach automatische Möglichkeiten zur Verfügung, wobei die Leistungsfähigkeit der automatischen Verfahren unterschiedlich ist. Dabei sollte darauf geachtet werden, daß Auswahlmöglichkeiten für die Einbeziehung von Bauelementen in einen automatischen Plazierungsvorgang bestehen. Denn bekanntlich gibt es bei jeder Konstruktion Bauelemente, deren Position vorgeschrieben ist – und wenn es sich dabei nur um den Steckverbinder handelt. Solche Bauelemente werden immer manuell vorplaziert. Außerdem lassen Systeme mit Auswahlmöglichkeiten der Bauelemente für die automatische Plazierung ein schrittweises Plazieren entsprechend den Bauelementekategorien zu.
Für die manuelle Plazierung gibt es in den einzelnen Systemen inzwischen recht wirkungsvolle Unterstützungsmöglichkeiten, wie unter anderem ein dynamisches Optimieren der Verbindungsbäume. Lassen Sie sich dazu aber vom Anbieter genau informieren. Die Verbindungsbäume werden meist als Luftlinien, die die logischen Verbindungen aus dem Stromlaufplan repräsentieren, dargestellt. Wurden die kritischen Bauelemente manuell plaziert, kann mit den automatischen Plazierungsroutinen weitergearbeitet werden. Um Irrtümern vorzubeugen, sei hier angemerkt, daß diese automatischen Routinen stets eine Empfehlung darstellen, in den meisten Fällen ist eine manuelle Korrektur des automatischen Plazierungsergebnisses notwendig. Dennoch sind solche Plazierungsroutinen eine wertvolle Hilfe, wenn eine Vielzahl gleicher Bauelemente (z. B. ICs) zu plazieren sind. Die Mehrzahl der Systeme verlangt vom Konstrukteur die Vorgabe einer Plazierungsmatrix. Das sind Linien in horizontaler und vertikaler Richtung, deren Abstände frei wählbar sind. Auf den Schnittpunkten dieser Linien werden die Bauelemente mit ihrem geometrischen Nullpunkt plaziert (Bild 6). Bei der Konstruktion und der Modifizierung dieser Matrizen sollte auf den angebotenen Komfort geachtet werden. Im allgemeinen erfolgt das Plazieren nach dem Kriterium der kürzesten Verbindungslängen. Daraus wird ersichtlich, daß mit fortschreitendem Plazierungsstand neue Konstellationen für die kürzesten Verbindungslängen entstehen. Deshalb sollte man sich über Optimierungsmöglichkeiten des Plazierungsstandes genau beraten lassen, weil hier teilweise große Unterschiede im Leistungsangebot bestehen. Wichtig scheint aus meiner Sicht das Vorhandensein eines Optimierungspasses bei der automatischen Plazierung für vollständig plazierte Leiterplatten durch Austausch der Bauelemente auf ihren Plätze, Gatter- und Pintausch, die Neuordnung der Verbindungsbäume und die Möglichkeit der Rückschreibung der Ergebnisse des Gatter- und Pintausches in den Stromlaufplan. Diese Algorithmen können sogar dazu führen, daß einzelne Bauelemente auf der Platte überflüssig werden. Zumindest aber sind sie eine wichtige Voraussetzung für eine möglichst

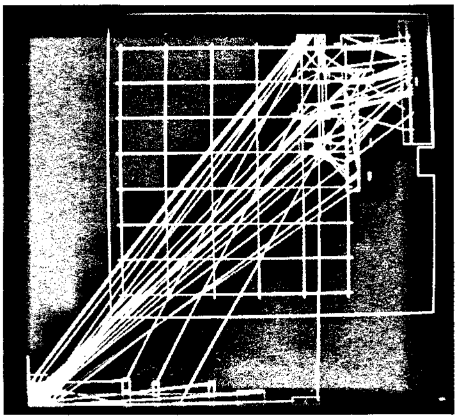


Bild 6 Leiterplatte mit Plazierungsmatrix während des Plazierens

komplikationsfreie Entflechtung der Verbindungen auf der Leiterplatte, insbesondere dann, wenn durch Verkürzung der Verbindungslängen auch gleich die „Kreuzung" von Verbindungslinien vermieden wird. Im Hinblick auf die Verwendung von SMD-Bauelementen sollte die automatische Spiegelung von Bauelementen und ihrer Pins bei der manuellen und automatischen Plazierung zur Selbstverständlichkeit gehören. In diesem Zusammenhang noch ein Hinweis: Alle Bauelemente der Konstruktion (Bauelementeumrisse, Pins, die grafische Darstellung der logischen Verbindungen, geroutete Trassen, Texte usw.) sollten in der grafischen Darstellung mit verschiedenen Farben dargestellt und auch unsichtbar gemacht werden können.

Gute Systeme routen selbständig

Ist ein befriedigender Plazierungsstand erreicht, kann die Wandlung der logischen Verbindungen in physische Trassen, das Routing, beginnen (Bilder 7 und 8). Auch hier bieten die Systeme manuelle und automatische Routinen an. Bereits bei den manuellen Routinen gibt es in der Handhabung erhebliche Unterschiede. Auf alle Vor- und Nachteile einzugehen, reicht der Platz in diesem Artikel nicht aus, wobei es der subjektiven Einschätzung eines jeden Konstrukteurs obliegt, einzuschätzen, ob eine Routine tatsächlich in der täglichen Arbeit von Vorteil ist. Allgemein kann aber gesagt werden, daß solche Arbeitsweisen wie „Gummibandtechnik", das teilweise Routen einer Verbindung, das partielle Löschen einer fertig gerouteten Trasse, Verschiebemöglichkeiten von Trassensegmenten (Stück von einer Trassenecke zur nächsten) und das automatische Setzen von Durchkontaktierungen als vorteilhaft angesehen werden und zum Standard gehören sollten.
Zunehmend spielt auch die Möglichkeit der Trassenverjüngung bei der Durchführung zwischen zwei Bauteilpins sowie die Wahl zwischen verschiedenen Arten von Durchkontaktierungen eine große Rolle. Unterschiedliche Auffassungen dagegen gibt es zur Entwurfsregelprüfung. Daß eine Prüfung des Entwurfs auf Einhaltung der eingestellten Konstruktionsparameter vorhanden sein muß, steht außer Zweifel. Einige Systeme prüfen beim manuellen Routing die Entwurfsregeln sofort (on-line checking); das kostet aber während der Arbeit Rechnerzeit, so daß dadurch das Antwortzeitverhalten ungünstig beeinflußt werden kann. Da man eine Trasse,

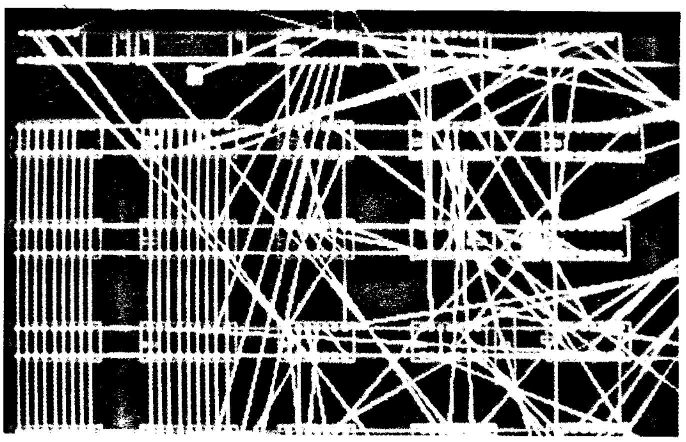


Bild 7 Detail der Leiterplatte vor dem Routen ...

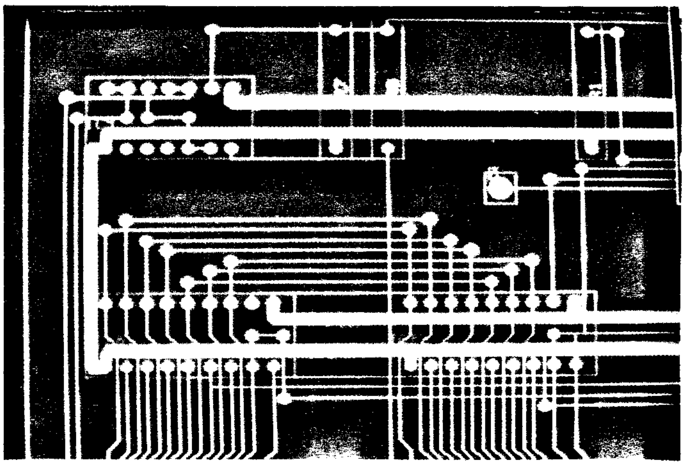


Bild 8 ... und nach dem Routen

die nicht den Entwurfsregeln entspricht, weil sie zum Beispiel irgendwo die Abstandswerte verletzt, nicht los wird, obwohl man sie erstmal nur ablegen will, um an einer kritischen Stelle Platz zu schaffen, bieten andere Systeme gesonderte Prüfroutinen an, die man bei jedem beliebigen Arbeitsstand und nach Abschluß der Arbeiten aufrufen kann. Optimal scheint deshalb eine Mischung aus beiden Verfahren zu sein.
Die automatischen Router auf PC-Basis sind inzwischen bei guten Systemen erstaunlich leistungsfähig, lassen aber insgesamt immer noch einige Wünsche offen. Das liegt meist daran, daß hierfür erhebliche Rechenleistungen notwendig sind, die sich auch im Zeitverhalten ausdrücken. Da ein Routerlauf von mehreren Stunden auf einem PC kaum akzeptabel ist, müssen also Kompromisse gefunden werden. Bei einigen Systemen wird ein Standardrouter angeboten, der in kurzer Zeit (je nach Packungsdichte und Anzahl der Verbindungen zwischen 5 und 15 Minuten) einen Trassierungsvorschlag liefert, der aber in den meisten Fällen manuell nachbehandelt werden muß. Für höhere Anforderungen werden dann zusätzlich Autorouter angeboten, die aber einen kräftigen Hardwarehintergrund brauchen, zumindest PCs mit 386er-Prozessoren. Hier sollte der künftige Anwender genau prüfen, ob sich die relativ hohe Investition (Hardware und Software) am Ende auszahlt.

Nur noch herstellen lassen?

Kommen wir jetzt zu einem kritischen Punkt im gesamten Konstruktionsprozeß. Die fertige Konstruktion auf dem Bildschirm soll jetzt in technologische Daten und Unterlagen umgesetzt werden. Die frühere Praxis der Leiterplattenhersteller bestand darin, eine erhebliche Menge verschiedener Unterlagen für die Leiterplattenherstellung anzufordern. Heute sollten jedoch auch die Hersteller der östlichen Bundesländer in der Lage sein, maschinenlesbare Datenträger mit den Steuerinformationen für Lichtzeichen- und Bohrmaschine entgegenzunehmen und daraus die Platine zu fertigen. Als Quasi-Standards haben sich dafür das Gerber-Format für die Lichtzeichenmaschinen und das Excellon-Format für die Bohrmaschinen durchgesetzt.
Um aber die Steuerdaten ableiten zu können, müssen die notwendigen Softwarehilfsmittel in den Entwurfssystemen enthalten sein. Informieren Sie sich bei Ihrem Anbieter sehr genau, wie leistungsfähig das angebotene System an dieser Stelle ist. Sie können durch eine gute Auswahl eine Menge Geld und Aufwand sparen. Empfehlenswert sind Systeme mit einer grafischen Unterstützung bei der Ableitung der Postprozeßdaten. Aber auch weitere Kriterien, wie die Möglichkeiten zum Skalieren, Verschieben und Drehen der konstruierten Platine, sollten Sie berücksichtigen. Fragen Sie ihren Anbieter auch, ob alle von Ihnen benötigten Fertigungsunterlagen ableitbar sind.

Die Hardware muß passen

Zum Schluß noch einiges zur Hardware. Ein AT-kompatibler PC mit 286er- oder 386er-Prozessor sollte selbstverständlich sein. Um das Entwurfssystem selbst, die Katalogdaten und alle im Entwurfsprozeß anfallenden Daten abspeichern zu können, ist in Abhängigkeit von der Anzahl der gleichzeitig zu bearbeitenden Projekte eine Festplatte von 40 bis 120 MByte erforderlich. Ein oder zwei Diskettenlaufwerke gehören ebenso zur Grundausstattung. Die Entwurfssysteme benötigen entweder ein Digitalisiertablett oder eine Maus für die Arbeit. Informieren Sie sich deshalb vor dem Softwarekauf darüber, damit Sie Ihre Hardware entsprechend bestellen können.
Wichtig ist die Auswahl des Grafikbildschirms. Auch hierfür sollten Sie sich von Ihrem Anbieter gründlich beraten lassen, denn im allgemeinen unterstützen die einzelnen Systeme immer nur einen Teil der insgesamt von den Hardware-Herstellern angebotenen Grafikbildschirme und -karten. Für den professionellen Einsatz sollten aber Auflösungen von 800 × 600 Bildpunkten (erweiterter VGA-Standard) die untere Grenze sein. Sparen Sie aber nicht an dieser Stelle, die Konstrukteure werden es Ihnen später danken.
Zur Grundausstattung sollte auch ein Plotter gehören. Auch hier sind die Angebote vielfältig. Günstig sind Plotter, die die HP-Sprachen verstehen; diese Ausgabe hat sich bei den Entwurfssystemen zum Quasi-Standard herausgebildet. Ebenso wird das Gerber-Format für die Lichtzeichentechnik fast ausnahmslos von allen Entwurfssystemen ausgegeben. Ob Sie sich allerdings die teure Lichtzeichentechnik kaufen wollen, hängt natürlich davon ab, ob in Ihrem Unternehmen gefertigt werden soll oder ob Ihr Hersteller Glasoriginale für die Produktion fordert. Ansonsten sollte man auf diese teure Investition verzichten.
Sollte bei Ihnen noch Lichtzeichentechnik älterer Produktion mit Lochstreifeneingabe im Einsatz sein, so werden Sie einen Lochstreifenstanzer am PC benötigen. Da aber ältere Lichtzeichenmaschinen und insbesondere die aus DDR-Produktion das Gerber-Format nicht verstehen, werden Sie Sprachwandlungsprogramme vom Gerber-Format in das Sprachformat Ihrer Lichtzeichenmaschine benötigen. Im allgemeinen kann der Software-Anbieter bei der Beschaffung solcher Programme vermittelnd helfen.
Für die Zukunft sind Sie immer gut beraten, wenn Sie die Möglichkeit der Vernetzung Ihres PCs bereits beim Kauf berücksichtigen. Und das aus zweierlei Gründen: Zum ersten werden Sie bald merken, daß eine PC-Station für Ihre Auftragsabwicklung nicht ausreicht. Um aber den Pflegeaufwand für die Kataloge nicht mehrfach betreiben zu müssen, sollte der PC als „Katalog-Server" ausgerüstet werden. Auch angeschlossene Plotter und andere Peripherie sind über ein solches Netz von allen PCs aus erreichbar, die Mehrfach-Investition solcher Geräte erübrigt sich.
Zum zweiten haben Sie die Möglichkeit, Ihren PC mit einer leistungsfähigen Workstation zu verbinden. Wenn Sie langfristig den Einsatz von Entwurfssoftware auf der Workstation beabsichtigen, noch dieser Hinweis: Wählen Sie ihre PC-Software auch unter dem Gesichtspunkt aus, daß Sie später die Workstation-Software vom gleichen Entwickler und damit mit der gleichen Systemphilosophie erwerben können. Sie sparen damit eine Menge Aufwand beim Datenaustausch zwischen PC und Workstation. Diese Verknüpfung ermöglicht es Ihnen beispielsweise, die Stromlaufpläne auf dem PC zu erfassen und einfache Entwürfe dort zu Ende zu bearbeiten. Komplizierte Entwürfe, die unter anderem höhere Router-Leistungen verlangen, können auf dem direkten Weg an die Workstation übergeben und dort fertiggestellt werden.
Auch wenn Ihre Geschäftslage heute solche Auftragsdichte und -kompliziertheit noch nicht erwarten läßt, sollten Sie sich dennoch der Zukunft nicht verschließen. Der Wechsel der Entwurfssysteme und deren Konzeption ist immer mit Problemen behaftet. Durchgängigkeit im Hardwarebereich und im Softwarebereich vom PC bis zur Workstation und umgekehrt bringt immer Vorteile, insbesondere bei den Kosten und den Einsatzmöglichkeiten.

✉ KONTAKT ☎

Systemhaus für Informationsverarbeitung GmbH, Jacques-Duclos-Straße 50, O-1156 Berlin

Schneller Logarithmieralgorithmus für Mikrorechner

Jörg Wunsch, Dresden

Die Logarithmierung binärer Zahlen ist ein in der Praxis häufig auftretendes Problem. Insbesondere zur grafischen Darstellung von Meßwerten und ähnlichem ist es oftmals notwendig, größere Datenmengen in möglichst kurzer Zeit zu logarithmieren. Dabei ist die erreichte Genauigkeit der üblichen Gleitkomma-Algorithmen im Grunde nicht notwendig, da die mögliche Darstellungsgenauigkeit der Grafik meist um einige Größenordnungen niedriger ist. Auch spielt die Basis des Logarithmus keine Rolle, da eine unterschiedliche Basis einer Multiplikation des Ergebnisses mit einem konstanten Faktor gleichkommt. Diese kann dann wiederum durch eine entsprechende Achseneinteilung korrigiert werden. Hingegen wirkt sich der große Zeitbedarf entsprechender Gleitkommaoperationen oft sehr nachteilig aus. Da Meßwerte zumeist ursprünglich als Festkommazahlen vorliegen, kann in solchen Fällen oft ein schnelles Logarithmierunterprogramm auf Maschinenniveau Abhilfe schaffen. Dieser Beitrag soll auf grundlegende Zusammenhänge dieses Problems eingehen und abschließend ein praktisches Beispiel anbieten. Die Beschränkung auf Festkommazahlen wird zwar von vornherein als gegeben angesehen, stellt aber kein grundlegendes Dogma dar.

Mathematische Grundlagen

Entsprechend der Zahlenbasis unserer Computer bietet sich für solche Logarithmieralgorithmen natürlich die Basis 2 zuerst an. Wenn man den ganzzahligen Anteil des Zweierlogarithmus einer Integerzahl i (nachfolgend als **ld i** bezeichnet) ermitteln will, so ist das Verfahren recht einfach: Man zählt die Bitposition aus, in der sich das höchstwertige Eins-Bit (in der Dualdarstellung) befindet. (Die Bitzählung erfolgt dabei in der üblichen Weise, das heißt, Bit 0 steht ganz rechts.) Zwei Beispiele:

i = 1234 = 010 011 010 010B
höchstwertige 1 in Bit 10,
d. h. int(ld(1234)) = 10;

i = 23 = 010 111B
Bit 4 enthält die erste Eins: int(ld(23)) = 4.

Der ganzzahlige Anteil des Logarithmus wird im allgemeinen selbst bei nur geringen Forderungen an die Auflösungsgenauigkeit nicht ausreichend sein: Wenn der Wertebereich von i $0 < i \leqq 65\,535$ ist, so ist der Wertebereich von x = ld(i) schließlich nur $0 \leqq x \leqq 15$. Leider sieht die Mathematik bei der Ermittlung des gebrochenen Anteils von ld i nicht mehr ganz so einfach aus.

Für die Ermittlung dieses gebrochenen Anteils (Mantisse) muß man schon etwas tiefer in die Mathematik einsteigen. Zunächst wird der nach obigem Prinzip ermittelte ganzzahlige Teil aus der weiteren Betrachtung ausgeklammert:

$$\mathbf{ld\,a = ld\,(b * 2^i) = i + ld\,b}, \qquad (1)$$

wobei
i ... ganzzahliger Anteil von ld a,
$1 \leqq b < 2$.

Die Zahl b ist somit 1 + <echt gebrochene Zahl>, auch darstellbar als:

$$\mathbf{b = 2^0 + c_1\,2^{-1} + c_2\,2^{-2} + \ldots + c_n\,2^{-n}}, \qquad (2)$$

$$\text{wobei } c_i \in \{0, 1\}. \qquad (3)$$

Der Koeffizient von 2^0 in Gleichung 3 ist das gefundene höchstwertige Eins-Bit und damit stets gleich eins. Die Koeffizienten c_1 bis c_n entsprechen den diesem Bit nachfolgenden Bitpositionen, von links numeriert. Damit wird das weitere Problem in die Logarithmierung der Zahl b verlagert. Diese läßt sich entsprechend den Logarithmengesetzen ohne weiteres auf eine andere Basis umrechnen mit:

$$\mathbf{x = ld\,b = \frac{ln\,b}{ln\,2} = k * ln\,b}, \qquad (4)$$

$$\text{wobei } k = \frac{1}{\ln 2} \approx 1{,}442695\,. \qquad (5)$$

Diese scheinbar umständliche Vorgehensweise bietet nun einen Vorteil: ln b kann für die Entwicklungsstelle 1 in eine Taylor-Reihe entwickelt werden, die folgendes Aussehen hat:

$$\ln b = \sum_{k=1}^{n} (-1)^{k-1} \frac{(b-1)^k}{k} \quad (n \to \infty) \qquad (6)$$

Dabei ergibt sich b − 1 entsprechend Gleichung (2) zu

$$\mathbf{b - 1 = c_1\,2^{-1} + c_2\,2^{-2} + \ldots + c_n\,2^{-n}}. \qquad (7)$$

Wenn man nun die Taylor-Reihe schrittweise auflöst, so ergibt sich eine Summe aus Koeffizienten der entsprechenden Zweierpotenzen 2^{-j}. Entsprechend dem Wertebereich der Koeffizienten c_i (Gleichung 3) bilden sich dabei die Potenzen dieser Koeffizienten auf sich selbst ab, das heißt

$$c_i^n = c_i \text{ (für } n > 0\text{, ganzzahlig).} \qquad (8)$$

Diese Rechnung kann man nach endlich vielen Schritten abbrechen, weil man sich eine Mantissengenauigkeit von beispielsweise maximal 1 Byte als Richtwert vorgeben kann. Dabei zeigt sich dann folgender verblüffender Effekt, der bei Betrachtung der Gleichungen 1 bis 6 augenscheinlich für unwahrscheinlich gehalten worden wäre:

$$\mathbf{ld\,b = c_1\,2^{-1} + c_2\,2^{-2} + c_3\,2^{-3} + (c_4 + (c_1 - c_2)^2)\,2^{-4} + (c_5 + (c_1 - c_3)^2)\,2^{-5} + \ldots} \qquad (9)$$

Rechentechnische Umsetzung

In Gleichung 9 lassen sich die Terme $(c_i - c_j)^2$ entsprechend Gleichung 3 auflösen zu:

$$\mathbf{(c_i - c_j)^2 = c_i\ .xor.\ c_j}\,. \qquad (10)$$

Somit lassen sich also die ersten fünf Mantissenbits von ld i mit guter Näherung nach folgenden Regeln ermitteln:

- Bit 1 (MSD) bis Bit 3 werden aus den Bitpositionen übernommen, die dem höchstwertigen Eins-Bit der Zahl i unmittelbar folgen (genannt c_1 bis c_3).
- Bit 4 ergibt sich zu c_4 + (c_1 .xor. c_2) sowie Bit 5 zu c_5 + (c_1 .xor. c_3), wobei Überträge in die vorderen Stellen möglich sind.
- Weitere Bitpositionen lassen sich leider nicht mehr so einfach ermitteln und werden deshalb bei Bedarf mit Nullen gefüllt.

Der damit realisierte Wertebereich beträgt zum Beispiel der bei Logarithmierung einer 16-Bit-unsigned-Zahl:

- 4 Bit für den ganzzahligen Teil (0...15)
- 5 Bit für die Mantisse.

Somit ergeben sich insgesamt 512 verschiedene Werte. Dies dürfte für den oben genannten Anwendungsfall (Grafik) meist ausreichend sein, wenn man den erzielbaren Rechenzeitvorteil berücksichtigt.

Bild 1 zeigt ein kurzes Z 80-Assemblerprogramm, das die angeführten Regeln praktisch umsetzt. Es wurde übersetzt und zu Testzwecken als Inline-Funktion in ein Turbo-Pascal-Programm eingebunden. In diesem wurde dann durch Vergleich mit den „genauen" Werten der Real-Rechnung für alle positiven ganzen Zahlen (2 bis 32767) der relative Fehler ermittelt. Es zeigten sich folgende Ergebnisse:
mittlerer relativer Fehler: 0,0020
Maximalwerte des relativen Fehlers:

Intervall	2..9	0,0093	bei i = 7
"	10..99	0,0091	bei i = 71
"	100..999	0,0092	bei i = 143
"	1000..9999	0,0074	bei i = 1151
"	10000..32767	0,0054	bei i = 10239

Dies bestätigt die entsprechenden bei der Auflösung der Taylor-Reihe gemachten Einschränkungen. Der vorgestellte Algorithmus wird deshalb also vorrangig dort gut anzuwenden sein, wo die Genauigkeitseinbuße zugunsten des Rechenzeitvorteils in Kauf genommen werden kann.

```
.z80
title dual logarithm of fix-point #
;
;input HL: unsigned #
;output D: fix part of ld(HL), 0<=x<16
;         E(3)..E(7): mantissa of ld(HL)
;         E(0)..E(2): 0
;
ld_of_HL::
        ld      a, h
        or      l
        jr      nz, ld1
;any error handling may occur here, e.g.:
        ld      de, 0
        scf
        jr      ldexit
;
ld1:    ld      b, 15
        ld      de, 8000h
ld2:    ;test for most significant 1-bit
        ld      a, h      ;test HL & DE
        and     d         ;jr   nz, ld3
        jr      nz, ld3
        ld      a, l
        and     e
        jr      nz, ld3
        srl     d         ;DE := DE / 2
        rr      e
        djnz    ld2
;
ld3:    ld      d, b      ;cycle counter: fix part
        ld      a, 16
        sub     b
        ld      b, a
ld4:    sla     l         ;shift c(1)..c(n) into
        rl      h         ;upper part of H
        djnz    ld4
;
        ld      a, h       ;copy mantissa bits
        and     11111000b ;c(1)..c(5)
        ld      e, a
        ld      a, h
        and     10000000b ;c(1) .xor. c(2)
        rrca
        push    af
        xor     h
        and     1000000b
        rrca
        rrca
        add     a, e
        ld      e, a
        pop     af
        rrca
        xor     h
        and     100000b ;c(1) .xor. c(3)
        rrca
        rrca
        add     a, e
        ld      e, a
        xor     a         ;clear cy --> succeeded
ldexit: ret
;
end
```

***Bild 1** Listing des vorgestellten Beispielprogramms zur schnellen Logarithmierung einer 16-Bit-Zahl*

Erweiterung für die Gleitkomma-Arithmetik

Das vorgestellte Verfahren eignet sich grundsätzlich auch zur näherungsweisen Logarithmierung von Gleitkommazahlen. Dabei vereinfacht sich der Ablauf sogar, weil Gleitkommazahlen rechnerintern in der Form

$$r = 2^a * b \text{ mit } 1 \leqq b < 2 \qquad (11)$$

abgespeichert werden, das heißt, der ganzzahlige Teil des Logarithmus ist bereits früher ermittelt worden. Er kann hier aber auch negativ sein, da Zahlen < 1 zugelassen sind. Meist wird er nicht in der sonst für vorzeichenbehaftete Zahlen üblichen Zweierkomplementdarstellung abgespeichert, sondern mit Offset 80H (bei 8-Bit-Exponenten) als sogenannte Charakteristik notiert. Zum Beispiel sind dann:

80H = Exponent 0, 1 = Exponent −127,
0FFH = Exponent 127, 0: Kennzeichnung der exakten Null

Die Zahl b, auch Mantisse der Zahl genannt (was nicht mit der Mantisse des Logarithmus zu verwechseln ist), besitzt, wie im obigen Fall, in der Bitposition 2^0 (entsprechend ihrem Wertebereich, s. Gleichung 11) stets ein Eins-Bit, was deshalb niemals abgespeichert wird (sogenanntes hidden bit). Dafür steht an dieser Stelle gewöhnlich das Vorzeichenbit für die gesamte Zahl. Dieses sollte vor dem Logarithmieren getestet werden (ebenso wie die Charakteristik 0), da der Logarithmus von Null oder einer negativen Zahl nicht definiert ist. Die nachfolgenden Bits entsprechen den in obiger Rechnung benutzten Bits c_1 bis c_n, wobei für das hier vorgestellte Näherungsverfahren wiederum c_1 bis c_5 herangezogen werden müssen.

✉ KONTAKT ☏
Jörg Wunsch, Deubener Straße 27, O-8028 Dresden

Grundlagen des logischen Programmierens Teil 3

Dr. Konrad Schultz, Chemnitz

Im dritten und letzten Teil geht es insbesondere darum, wie mit Hilfe eines logischen Programms Fragen zu beantworten sind. Das erfordert Kenntnisse, was logisches Folgern heißt und wie eine Folgerungsbeziehung festzustellen ist.

Die Skolemschen Normalformen

Angenommen, wir haben ein Modell einer Aussage φ. Dieses Modell ist dann zugleich ein Modell jeder Aussage $\exists X\varphi$(name/X). (*φ(name/X) bezeichne dabei wieder die Aussagenform, die aus φ* dadurch hervorgeht, daß die Individuenkonstante *name* überall durch die Variable X ersetzt wird.) Wenn wir ein Modell einer Aussage $\forall Y\varphi$ haben, so haben wir ebenso zugleich ein Modell jeder Aussage $\forall Y\exists X\varphi$(fname(Y)/X), wobei *fname* eine einstellige Funktionskonstante ist. Umgekehrt lassen sich Modelle der partikularisierten Aussagen zu Modellen von φ oder $\forall Y\varphi$ fortsetzen, indem *name* bzw. *fname* geeignet interpretiert wird. Daraus folgt:

Die Aussagen $\exists X\psi$ bzw. $\forall Y\exists X\psi$ sind genau dann logisch erfüllbar (d. h., sie besitzen ein Modell), wenn für eine jeweils neue (noch nicht vorkommende) Individuenkonstante a bzw. Funktionskonstante f die Aussagen ψ(X/a) bzw. $\forall Y\psi$(X/f(Y)) logisch erfüllbar sind.

Auf diese Weise kommt man zu einer **erfüllbarkeitstheoretischen Skolemschen Normalform:**

Man ersetze in einer pränexen Normalform φ gleich $\forall X_1 \ldots \forall X_{i-1}\exists X_i\varphi'$ (φ' darf noch mit Quantifizierungen beginnen) die Variable X_i durch den Term $f(X_1,\ldots,X_{i-1})$ für eine neue Funktionskonstante f und lasse $\exists X_i$ weg. Dieses Verfahren werde so oft wie möglich wiederholt. Das Ergebnis ist eine (erfüllbarkeitstheoretische) Skolemsche Normalform ψ, eine pränexe Normalform ohne $\exists$, die genau dann logisch erfüllbar ist, wenn φ logisch erfüllbar ist.

Ebenso kommt man zu einer **allgemeingültigkeitstheoretischen Skolemschen Normalform:**

Man ersetze in einer pränexen Normalform φ gleich $\exists X_1 \ldots \exists X_{i-1}\forall X_i\varphi'$ die Variable X_i durch den Term $f(X_1,\ldots,X_{i-1})$ für eine neue Funktionskonstante f und lasse $\forall X_i$ weg. Das Verfahren werde so oft wie möglich wiederholt. Das Ergebnis ist eine (allgemeingültigkeitstheoretische) Skolemsche Normalform ψ, eine pränexe Normalform ohne $\forall$, die genau dann allgemeingültig ist, wenn φ allgemeingültig ist.

Wenn nun Φ eine Menge von Aussagen ist und Φ' dadurch aus Φ hervorgeht, daß alle Aussagen durch dazugehörige erfüllbarkeitstheoretische Skolemsche Normalformen ersetzt werden, so gilt wiederum: Φ ist genau dann logisch erfüllbar (besitzt ein Modell), wenn Φ' logisch erfüllbar ist, bzw. genau dann, wenn Φ' ein kanonisches Modell besitzt.

Sei weiter Φ'^{o} die Menge aller Grundbeispiele von Elementen von Φ'. Ein kanonisches Modell von Φ' ist offenbar auch ein kanonisches Modell von Φ'^{o} und umgekehrt. Somit ist Φ genau dann logisch erfüllbar, wenn Φ'^{o} ein kanonisches Modell besitzt.

Nun gilt für einfache Aussagen φ die Beziehung $\Phi \models \varphi$ genau dann, wenn es kein Modell von $\Phi \cup \{\neg\varphi\}$ gibt, und damit genau dann, wenn es kein kanonisches Modell von $\Phi'^{o} \cup \{\neg\varphi\}$ gibt, und damit auch genau dann, wenn gilt $\Phi'^{o} \models \varphi$.

Hinzu kommt ein weiterer Satz aus der mathematischen Logik: Falls Ψ eine (eventuell unendliche) Menge von Aussagen ist (zum Beispiel Ψ gleich Φ'^{o}) und gilt $\Psi \models \psi$, so gibt es eine endliche Teilmenge Ψ^* von Ψ, für die auch schon $\Psi^* \models \psi$ gilt.

Um nachzuweisen, daß gilt $\Phi \models \varphi$, braucht der Computer also nur eine endliche Menge Φ'^{o*} von Grundbeispielen aus Φ'^{o} zu finden mit $\Phi'^{o*} \models \varphi$. Und das kann der Computer grundsätzlich. Die endlichen Teilmengen von Φ'^{o} sind für den Computer (theoretisch) beherrschbar, und ob aus ihnen φ folgt, kann er überprüfen.

Falls gilt $\Phi \models \varphi$, kann der Computer das (theoretisch) also auf jeden Fall beweisen und dann beispielsweie in Prolog antworten

yes

und eventuell auch noch einen Beweis dazu aufschreiben.

Erhalten wir vom Prolog-System trotzdem keine Antwort auf unsere Frage, so liegt das meistens an einem Programmierfehler, wenn der Programmierer die spezielle Ablauforganisation von Prolog etwa bei der Art der Formulierung rekursiver Eingaben nicht berücksichtigt hat und den Computer damit unnötigerweise in Endlosschleifen oder sonst in die Irre schickt. Falls allerdings nicht gilt $\Phi \models \varphi$, kommt zwar bestimmt kein *yes*, aber wir sind der Antwort *no* auch nicht sicher. Der Computer kann in diesem Falle in eine unendliche Schleife geraten. Es ist sogar so, daß wir die erforderliche Rechenzeit des Computers für ein eventuelles *yes* grundsätzlich nicht abschätzen können. Es kann sein, daß er schon 50 000 Jahre nach einer endlichen Menge von Grundbeispielen aus Φ'^{o} gesucht hat, wir dann aber durchaus nicht wissen, ob wir bloß nicht lange genug auf das *yes* gewartet haben, oder ob es deshalb nicht kommt, weil $\Phi \models \varphi$ gar nicht gilt. Das ist gleichbedeutend damit: Es kann keinen Computer geben, der für beliebige φ entscheiden kann, ob $\Phi \models \varphi$ oder nicht $\Phi \models \varphi$.

In den Anfangszeiten der Logikprogrammierung hat man die Computer systematisch alle endlichen Mengen von Grundbeispielen aufbauen und die aufgebauten Mengen testen lassen; in der Zwischenzeit ist man auf effizientere Verfahren gekommen (siehe Abschnitt Resolutionsverfahren).

Fragen nach Variablenwerten

Bei der Frage nach $\Phi \models \varphi$, also der Frage

?-φ.

bezüglich einer „Wissensbasis" Φ aus Klauseln haben wir bisher nur einfache Aussagen φ betrachtet. Wir setzen jetzt nur noch voraus, daß φ in allgemeingültigkeitstheoretischer Skolemscher Normalform $\exists X_1 \ldots \exists X_k\, \varphi_o$ vorliegt. Die Elemente von Φ seien dagegen in erfüllbarkeitstheoretischer Skolemscher Normalform gegeben. Da $\neg\varphi$ dann logisch äquivalent zu einer erfüllbarkeitstheoretischen Skolemschen Normalform ist und es nur darauf ankommt, ob $\Phi \cup \{\neg\varphi\}$ logisch erfüllbar ist, sind die Voraussetzungen legitim.

$\Phi \cup \{\neg\varphi\}$ besitzt genau dann kein Modell, wenn wiederum eine endliche Menge von Grundbeispielen kein Modell besitzt, also eine endliche Menge Φ^{o*} von Grundbeispielen für Elemente von Φ und von Grundbeispielen $\neg\varphi_1,\ldots,\neg\varphi_n$ für $\neg\varphi$ kein Modell besitzt, demzufolge logisch widerspruchsvoll ist. Das ist aber gleichwertig mit $\Phi^{o*} \models \varphi_1 \vee \ldots \vee \varphi_n$. Die Grundbeispiele φ_i haben dann die Gestalt $\varphi_o(X_1/t_{1i},\ldots,X_k/t_{ki})$ für geeignete Grundterme $t_{1i},\ldots,t_{ki}$.

Es ergibt sich: Wenn $\Phi \models \exists X\, \psi$, so liefert die Logikprogrammierung (und damit der (theo-

retische) Computer) Grundterme $t_1,...,t_n$ mit $\Phi \models \psi(X/t_1) \vee ... \vee \psi(X/t_n)$. Für n = 1 wäre die Lösung konstruktiv, das ist aber nicht allgemein zu erreichen. In Prolog allerdings zahlen sich die folgenden Einschränkungen aus:

① Die Elemente von Φ sind (meist generalisierte) Hornklauseln (Fakten und Regeln) und damit auch in erfüllbarkeitstheoretischer Skolemscher Normalform.

② $\exists X\psi$, die Frageaussage (in Prolog Zielklausel genannt), ist eine partikularisierte Konjunktion (die Partikularisierung wird üblicherweise ebensowenig mitgeschrieben wie die Generalisierung in den Horn-Klauseln) von einfachen Aussagen und steht damit in allgemeingültigkeitstheoretischer Skolemscher Normalform.

Dann ist n = 1, also $\Phi \models \exists X\psi$ zieht nach sich $\Phi \models \psi(X/t)$ für einen Grundterm t, und diese konstruktive Lösung t wird vom Computer geliefert. Der Computer ist in Prolog sogar darüber hinaus in der Lage, sämtliche „Belege" zu liefern, auf die eventuell systemabhängig modifizierte Frage

?-liebt(hans,X),liebt(X,hans).

(also inhaltlich: Bestimme die Terme t mit

$\Phi \models$ liebt(hans,t) $\wedge$ liebt(t,hans))

kommt bei geeignetem Φ dann zum Beispiel die Antwort

X = grete

X = mutter_von(hans)

und vielleicht noch ein Vermerk, daß weitere Werte von X nicht zu erwarten sind. Bei manchen Prolog-Systemen kommt auch zunächst nur eine Antwort, weitere Antworten können dann durch ; abgefordert werden.

Zum Resolutionsverfahren

Das Resolutionsprinzip braucht ein Prolog-Programmierer dann nicht unbedingt zu kennen, wenn er dem System alle Klauseln (Fakten und Regeln) logisch exakt mitteilt und dies obendrein in der richtigen Reihenfolge tut. Wenn aber Fehler vorkommen, das System nicht so reagiert, wie man es sich vorgestellt hat (wie wohl zu Beginn meistens), dann ist schon eine gewisse Kenntnis der Prolog-Arbeitsorganisation erforderlich, um die Auswirkungen von Fehlern beurteilen und umgekehrt von den Auswirkungen auf die verursachenden Fehler schließen zu können. Ich stelle in diesem Abschnitt dar, wie der Computer logisch schließt, und komme damit auf das Resolutionsverfahren.

Da $\Phi \models \varphi$ genau dann gilt, wenn $\Phi \vee \{\neg\varphi\}$ logisch widerspruchsvoll ist, bemüht man sich in der Logikprogrammierung meist um den Nachweis der logischen Widersprüchlichkeit gewisser Aussagenmengen Φ. Wir setzen voraus, daß alle Elemente von Φ erfüllbarkeitstheoretische Skolemsche Normalformen sind und die Generalisierungen weggelassen sind. Die Reste seien als konjunktive Normalformen gegeben. Dann dürfen wir auch voraussetzen, daß die Elemente von Φ Alternativen von Literalen sind, Konjunktionen also in ihre Bestandteile aufgespalten sind.

Ein logischer Widerspruch in Φ kann sich höchstens darin markieren, daß ein Beispiel eines Gliedes ψ_1 einer Alternative $\varphi_1 \in \Phi$ gerade die Negation eines Beispiels ψ_2 einer anderen Alternative $\psi_1 \in \Phi$ ist. Von einer Alternative ψ zu einem Beispiel überzugehen, bedeutet die Anwendung einer Termeinsetzung σ auf φ, das Ergebnis sei mit $\sigma(\varphi)$ bezeichnet. Im Beispiel trifft also zu $\sigma_1(\psi_1)$ gleich $\neg\sigma_2(\psi_2)$. (Wir sagen, daß ψ_1 und ψ_2 **unifizierbar** sind, identifizierbar durch passende Einsetzungen in die Variablen.) Wenn nun φ_1' bzw. φ_2' dadurch entsteht, daß alle (evtl. mehrere) Glieder $\sigma_1(\psi_1)$ in $\sigma_1(\varphi_1)$ und alle Glieder $\sigma_2(\psi_2)$ in $\sigma_2(\varphi_2)$ weggelassen werden, so nehmen wir die Alternative $\varphi_1' \vee \varphi_2'$ zu Φ hinzu. Durch die Hinzunahme bleibt ein widerspruchsvolles Φ widerspruchsvoll, ein widerspruchsfreies Φ bleibt aber widerspruchsfrei, denn $\sigma_1(\varphi_1)$ ist logisch äquivalent zu $\neg\varphi_1' \rightarrow \sigma_1(\psi_1)$ und $\sigma_2(\varphi_2)$ ist logisch äquivalent zu $\neg\sigma_2(\psi_2) \rightarrow \varphi_2'$ gleich $\sigma_1(\psi_1) \rightarrow \varphi_2'$, und $\varphi_1' \vee \varphi_2'$, logisch äquivalent zu $\neg\varphi_1 \rightarrow \varphi_2$, folgt logisch aus beiden und damit aus $\{\varphi_1, \varphi_2\}$ und aus Φ. Wenn wir das Verfahren zu Ende führen, ergibt sich:

Φ ist genau dann logisch widerspruchsvoll, wenn das beschriebene sog. Resolutionsverfahren schließlich zu einer leeren Alternative führt.

Im Falle von Prolog mit seinen Horn-Klauseln ergibt sich nach der gleiche Methode folgendes: Von zwei Klauseln φ_o:- φ_1, φ_2, ... und ψ_o:– ψ_1, ψ_2, ... mit etwa $\sigma_2(\psi_o) = \sigma_1(\varphi_1)$ geht man über zu einer Klausel $\sigma_1(\psi_o)$:–$\sigma_2(\psi_1)$, $\sigma_2(\psi_2)$,...,$\sigma_1(\varphi_2)$,$\sigma_1(\varphi_3)$,... . und fügt diese zu Φ hinzu. Dies läßt sich verhältnismäßig anschaulich wie folgt umschreiben: Die negativen Literale sind die rechts von :–, die positiven Literale die links von :– (wegen a:–b,...,c≡¬b∨...∨¬c∨a). Die Fakten sind positiv, die Glieder der Fragen negativ. Ein negatives Literal (insbesondere der Frage) wird (erfolgreich) aufgerufen, indem ein Beispiel dieses Literals mit einem Beispiel eines positiven Literals zur Deckung gebracht wird (die beiden Literale werden unifiziert) und das Beispiel dieses positiven Literals (erfolgreich) aufgerufen wird. Der Aufruf des positiven Literals eines Fakt-Beispiels ist immer unmittelbar erfolgreich. Das positive Literal eines Regel-Beispiels wird (erfolgreich) aufgerufen, indem alle negativen Literale desselben Regel-Beispiels (erfolgreich) aufgerufen werden. Die Frage wird mit *yes* oder den Variablenwerten beantwortet, falls alle Glieder erfolgreich aufgerufen wurden.

Das Resolutionsverfahren liefert in jedem Falle die Lösungen, die Variablenwerte, mit. Ich erkläre das nun an meinem Prolog-Beispiel: Mit jedem Aufruf eines negativen Literals werden gewissen Variablenbindungen eingeführt, es wird nämlich eine (möglichst allgemeine) Substitution gemacht, die gewissen Variablen gewisse Terme zuordnet (Übergang vom Literal zu einem Beispiel). Diese Zuordnung, Bindung genannt, bleibt so lange gültig, bis der verursachte Aufruf sich als erfolglos herausstellt. Dann jedoch, wenn wir also in eine Sackgasse geraten sind, wird ein Zurück – ein **Backtracking** – ausgelöst, die Variablenbindung wieder aufgehoben und nach Möglichkeit mit einem anderen positiven Literal unifiziert, was eine neue, alternative, Variablenbindung auslöst. Falls das nicht möglich ist, ist der jeweils vorher versuchte Aufruf (der seinerseits das betrachtete negative Literal aufgerufen hat) ebenfalls erfolglos, und das Backtracking muß fortgesetzt werden. Die Variablenbindungen, die zum Schluß übrigblieben (das kann nur beim erfolgreichen Aufruf aller Frageglieder sein), liefern die gesuchten Variablenwerte.

Es sei noch bemerkt, daß sich die Resolutionsmethode im Falle von Horn-Klauseln wesentlich effizienter gestalten läßt, es sind weniger Unifizierungen zu versuchen, Doppelarbeit und Backtracking sind seltener.

Abseits der „reinen" Logikprogrammierung in Prolog

Ich hatte schon bemerkt, daß es in Prolog wie in jeder Sprache der Logikprogrammierung Strukturen gibt, die nicht nur oder am allerwenigsten als Aussagen zu interpretieren sind, Ausgabeanweisungen etwa, die zwar als *wahr*, also als erfolgreich aufrufbar gelten, bei denen es aber doch in erster Linie auf sogenannte Seiteneffekte ankommt, bei denen also die prozedurale Interpretation die entscheidende ist. Hier nimmt es nicht wunder, daß die Reihenfolge der Atome in den Klauseln bzw. die Reihenfolge der Klauseln von Bedeutung ist, wie wir bereits gesehen haben.

Es kommt aber auch im Bereich primär deklarativer Interpretationen auf Reihenfolge- und andere organisatorische Fragen an. Wenn wir etwa definieren

fak(0,1)

fak(X,Y):– A is X-1, fak(A,B), Y is B*X.

– wobei ein Atom *Variable is <Term>* (konsequenter wäre die Schreibweise *is(Variable,<Term>)* realisiert wird, indem der Wert von <Term> der Variablen *Variable* zugewiesen wird –, erhalten wir auf die Frage

?-fak(3,X)

ganz richtig die Antwort

X = 6

Wenn wir aber die beiden Klauseln bei der Eingabe in ihrer Reihenfolge vertauschen, so kommt auf die Frage

?-fak(3,X)

keine Antwort, weil der Aufruf von fak(M,X) nacheinander fak(2,X1), fak(1,X2), fak(0,X3), fak(-1,X4), ... aufruft ohne Ende. Prolog ruft nämlich die positiven Klauselköpfe stereotyp von oben nach unten und die negativen Atome der einzelnen Klauseln ebenso stereotyp von links nach rechts auf. Das ist eine dem Wesen nach algorithmische Arbeitsweise, dem Computer angemessen, aber nicht der Logik. An parallelen Konzepten, die solche Schwierigkeiten vermeiden, wird intensiv gearbeitet.

Es gibt ein recht umstrittenes Konzept in Prolog, das Schnittprädikat (den *Cut*) !, mit dem der Programmierer in die sonst starre Prolog-Abarbeitungsorganisation eingreift. Ein Aufruf von ! ist grundsätzlich erfolgreich, aber hier kommt es auf einen Seiteneffekt an: Einige Variablen dürfen bei einem eventuellen Backtracking kein zweites Mal gebunden werden. Kein Beispiel einer Regel

φ_o:– φ_1, ..., φ_k, !, φ_{k+1}, ...

darf im Zuge eines Backtrackings über ! hinaus zurück durch ein alternatives Beispiel ersetzt werden. Es wird also eine ganze Reihe von sonst möglichen alternativen Variablenbindungen abgeschnitten, was viel Rechenzeit sparen kann. Zuweilen wird die Verwendung von ! jedoch zur Programmiermethode. Ein Ziel der Logikprogrammierung, die Konzentration auf „Programme", in denen nur die reine Logik des Problems dargestellt ist, und die deshalb leicht verständlich und leicht korrigierbar sind, wird verlassen zugunsten algorithmischer Eingriffe, die das Programm zwar schneller machen können, aber die Transparenz nehmen.

Prolog verfügt über keine Negation, Motive dafür haben wir schon genannt: Insbesondere kommt durch die Hinzunahme der klassischen Negation ein nichtkonstruktiver Zug in die Logikprogrammierung (es gibt aber auch konstruktive Negationsformen, die in der Logikprogrammierung noch keinen allge-

meinen Eingang gefunden haben), und die Arbeitsweise des Logik-Computers wird wesentlich komplizierter. Im übrigen reicht der Bereich der Horn-Klauseln in folgender Beziehung aus: Wenn zwischen verschiedenen Größen ein algorithmischer Zusammenhang besteht, so läßt er sich durch ein Prolog-Programm beschreiben und auch realisieren. Das ist die sogenannte algorithmische Vollständigkeit von Prolog.

Es soll aber noch auf zwei negationsähnliche Erscheinungen in Prolog hingewiesen werden. Die eine ist das Standardprädikat *fail*. *fail* ist eine nullstellige Prädikatenkonstante, die immer als falsch gilt, als nie realisierbar (nicht erfolgreich aufrufbar). Logisch gilt

$\neg\varphi \equiv \varphi \rightarrow \text{fail}$

für beliebige Aussagen φ, aber eine entsprechende Regel

fail:$-\varphi$.

ist in Prolog nicht zugelassen, formal, weil der Regelkopf kein Standardprädikat wie *fail* sein darf. Inhaltlich ist der Aufruf von *fail* per definitionem zum Scheitern verurteilt und kann nicht erst vom Scheitern des Aufrufs von φ abhängig gemacht werden.

Der andere Negationsersatz ist *not*. Ein Aufruf von not(φ) ist erfolgreich, falls der Aufruf von φ in endlicher Zeit scheitert. Dieses sogenannte *negation by failure* ist der nächstliegende Ersatz für $\neg$, aber eben nur ein Ersatz. Falls nämlich der Aufruf von φ den Computer in eine unendliche Schleife führt, ist auch der Aufruf von not(φ) erfolglos, obgleich $\neg\varphi$ dann wahr wäre. Wenn in φ eine Variable X vorkommt, so wird die Frage

?-not(φ)

mit *yes* beantwortet, falls die Annahme $\Phi \models \exists X \varphi$ in endlich vielen Schritten widerlegt werden kann. Dann gilt $\forall X \neg\varphi$ in einem Modell von Φ. Das *not* bringt so eine Unsymmetrie bezüglich der Variablenbehandlung. Während eine Frage

?-alpha(beta(X)).

Antworten der Art

X=name

liefert, liefert die syntaktisch äquivalente Frage

?-not(beta(X)).

keine Variablenwerte, höchstens ein *yes* oder *no*. *?-alpha(beta(X)).* wird als Frage nach $\Phi \models \exists X$ alpha(beta(X)) verstanden, *?-not(beta(X))* als Frage nach

$\Phi \models$ not($\exists X$ beta(X)) und nicht etwa nach

$\Phi \models \exists X$ not(beta(X)).

Schluß

✉ KONTAKT ☎

Technische Universität Chemnitz, **Sektion Informatik**, Postfach 964, O-9010 Chemnitz; Tel. 66 82 52

Komfortable Funktionsberechnung mit Turbo-Pascal

Bernd Matzke, Delitzsch

*In der Programmiersprache Pascal steht zur Umwandlung einer in Form eines Strings vorliegenden Zahl in einen numerischen Wert die Prozedur **Val** zur Verfügung. Diese Funktion wandelt aber stets nur eine einzige Zahl um, die Behandlung zusammengesetzter mathematischer Ausdrücke oder Standardfunktionen (wie z. B. innerhalb einiger Basic-Dialekte) ist nicht möglich.*

*Da es aber in vielen Fällen wünschenswert wäre, derartige zusammengesetzte Ausdrücke berechnen zu können (Tischrechner, Tabellenkalkulation, Eingabe-Bearbeitung ähnlich /1/ usw.), soll im folgenden ein komfortabler Ersatz für die Funktion **Val** vorgestellt werden.*

Die Behandlung von Formeln erfordert die Aufspaltung des Ausdrucks in einzelne Teile, die dann durch die Standard-Prozedur **Val** behandelt werden können. Die so erhaltenen numerischen Zwischenwerte sind anhand der Operationszeichen miteinander zu verknüpfen.

Diese Angaben erfüllt die Funktion **Ergebnis**; ihr wird ein mathematischer Ausdruck als String übergeben, zurückgegeben wird der numerische Wert des Ausdruckes als **Real**-Zahl. Bei der Bildung des Ausdruckes müssen allerdings einige Regeln berücksichtigt werden:

Argumente von Funktionen müssen stets in Klammern gesetzt werden.

Es dürfen nie zwei Operationszeichen unmittelbar aufeinander folgen. Wird das Minuszeichen als Vorzeichen verwendet, so ist die zu negierende Zahl mit dem Vorzeichen in Klammern zu setzen, also 2∗(−3) anstatt 2∗−3. Erlaubt ist hingegen ein führendes Minus am Anfang eines (Teil-) Ausdruckes, z. B. 2^(−3∗6) oder −2∗3

Die Exponentialschreibweise (1e4 statt 10000) darf ebenfalls verwendet werden.

Innerhalb der Funktion wird versucht, den Ausdruck in zwei Teile (die Operanden) und ein beide Operanden verbindendes Operationszeichen aufzuspalten, wobei die Klammerung und die Vorrangregeln der Operationszeichen untereinander zu berücksichtigen sind. Gelingt dies, so werden die numerischen Werte der beiden Operanden entsprechend dem Operationszeichen miteinander verknüpft. Die Definition eigener Operationszeichen ist möglich (z. B. "^" für Potenzierung, siehe Bild 1). Da die beiden Operanden natürlich auch komplexe Ausdrücke sein können, sind sie wiederum mit der rekursiven Funktion **Ergebnis** zu behandeln.

Ist das weitere Aufspalten eines Ausdruckes nicht mehr möglich, so ist aus diesem unteilbaren Ausdruck der entsprechende numerische Wert zu bestimmen.

Das Auftreten eines nicht weiter zerlegbaren Ausdruckes stellt also vorerst die Abbruchbedingung für die Rekursion dar. Der so erhaltene elementare Ausdruck kann eine durch die Funktion **Val** verarbeitbare Zahl oder aber ein Funktionsaufruf sein, der mit **Val** noch nicht behandelt werden kann. Der Ausdruck muß also weiter bearbeitet werden. Diese Aufgabe übernimmt die Funktion **Wert.** Ergibt die Behandlung des Ausdrucks mit **Val** einen

```
 1: PROGRAM Val;
 2: {$A-}                                          {Rekursion erlauben}
 3: {$V-}                                     {keine Strinlaengenpruefung}
 4: CONST max_funkt = 10;
 5:       buchstaben: SET OF CHAR = ['A'..'Z'];
 6:       op_zeich  : SET OF CHAR = ['+','-','*','/','^'];
 7:       funkt_name: ARRAY[1..max_funkt] OF STRING[6] =
 8:         ('ARCTAN','COS', 'EXP','LN', 'SIN',
 9:          'SQR',   'SQRT','RAD','DEG','PI');
10:
11: TYPE string_typ = STRING[60]; {String fuer Formel, Laenge nach Bedarf}
12:      s10        = STRING[10];
13:      var_zeiger = ^var_rec;
14:      var_rec    = RECORD                 {Element fuer Variablenliste}
15:                     name  : s10;           {Bezeichnung der Variablen}
16:                     zeiger: ^REAL;              {Zeiger auf Variable}
17:                     next  : var_zeiger;  END;
18:
19: VAR  s: string_typ;
20:      var_anfang,var_akt: var_zeiger;                    {Variablenliste}
21:      n,fehler: INTEGER;
22:      x,y:  REAL;
23:
24: PROCEDURE valn(formel:string_typ; VAR x:REAL; VAR fehler:INTEGER);
25: {Ersatz fuer Standard-Funktion VAL}
26: VAR help:string_typ;
27:     n:INTEGER;
28:
29:   FUNCTION ergebnis(formel:string_typ):REAL; FORWARD;
30:
31:   FUNCTION nummer(VAR f_name:string_typ):INTEGER;
32:   {suchen Funktionsbezeichnung in Liste, Rueckgabe der Ordnungsnr.}
33:   VAR n:INTEGER;
34:   BEGIN
35:     nummer:=0;         {Rueckgabewert, wenn Bezeichnung nicht in Liste}
36:     FOR n:=1 TO max_funkt DO
37:       IF funkt_name[n] = f_name THEN nummer:=n;
38:   END;
39:
```

```
 40:   FUNCTION opz_pos(VAR formel:string_typ):INTEGER;
 41:   {Position letztes Operationszeichen mit niedrigster Wertigkeit }
 42:   {ausserhalb von Klammern}
 43:   VAR ebene,wertigkeit,n:INTEGER;
 44:   BEGIN
 45:     opz_pos:=0;              {Rueckgabewert, wenn kein Zeichen gefunden}
 46:     ebene:=0;
 47:     wertigkeit:=MaxINT;
 48:     FOR n:=1 TO Length(formel) DO
 49:       CASE formel[n] OF
 50:            '(' :ebene:=ebene+1;
 51:            ')' :ebene:=ebene-1;
 52:         '-','+' :IF (ebene = 0) THEN    {wegen Exponetialschreibweise}
 53:                    IF (n=1) OR (n>1) AND (formel[n-1] <> 'E') THEN
 54:                      BEGIN opz_pos:=n; wertigkeit:=1; END;
 55:         '/','*' :IF (ebene = 0) AND (wertigkeit >1) THEN BEGIN
 56:                    opz_pos:=n; wertigkeit:=2; END;
 57:            '^' :IF (ebene = 0) AND (wertigkeit >2) THEN BEGIN
 58:                    opz_pos:=n; wertigkeit:=3; END; END;
 59:  END;
 60:
 61:   PROCEDURE klammern_entfernen(VAR f:string_typ);
 62:   {Klammern um gesammte Formel entfernen, wenn vorhanden}
 63:   BEGIN
 64:     IF ( opz_pos(f)=0 ) AND ( f[1] = '(' ) THEN
 65:       IF f[ Length(f)] = ')' THEN f:=Copy( f,2,Length(f)-2 )
 66:                              ELSE fehler:=1;        {kein Klammerpaar}
 67:   END;
 68:
 69:   PROCEDURE formel_aufspalten(VAR formel,op1,opz,op2:string_typ);
 70:   VAR posit:INTEGER;
 71:   BEGIN
 72:     posit:=opz_pos(formel);
 73:     IF posit <> 0 THEN BEGIN
 74:       op1:=Copy( formel,1,posit-1 );
 75:       klammern_entfernen(op1);
 76:       IF (op1<>'') AND (op1[ Length(op1)] IN op_zeich) THEN fehler:=2;
 77:       opz:=Copy(formel,posit,1);
 78:       op2:=Copy(formel,posit+1,Length(formel));
 79:       IF (op2<>'') AND (op2[1] IN op_zeich) THEN fehler:=2; END
 80:     ELSE BEGIN
 81:       op1:=formel;
 82:       klammern_entfernen(op1);
 83:       opz:=''; op2:=''; END;
 84:   END;
 85:
 86:   FUNCTION potenz(basis,exponent:REAL):REAL;
 87:   BEGIN
 88:     IF basis>0 THEN potenz:=Exp( exponent * Ln(basis) )
 89:                ELSE fehler:=3;
 90:   END;
 91:
 92:   FUNCTION wurzel(VAR f:string_typ):REAL;
 93:   VAR x:REAL;
 94:   BEGIN
 95:     x:=ergebnis(f);
 96:     IF x >= 0 THEN wurzel:=Sqrt(x)
 97:               ELSE fehler:=6;
 98:   END;
 99:
100:   FUNCTION lnn(VAR f:string_typ):REAL;
101:   VAR x:REAL;
102:   BEGIN
103:     x:=ergebnis(f);
104:     IF x > 0 THEN lnn:=Ln(x)
105:              ELSE fehler:=3;
106:   END;
107:
108:   FUNCTION v_wert(VAR v:string_typ):REAL;
109:   {suchen der Variablen mit dem uebergebenen Namen in der vom }
110:   {Nutzer angelegten Variablenliste und Rueckgabe ihres Inhaltes}
111:   VAR e:var_zeiger;
112:   BEGIN
113:     e:=var_anfang;
114:     WHILE (e <> NIL) AND (e^.name <> v) DO e:=e^.next;
115:     IF e^.name = v THEN v_wert:=e^.zeiger^
116:                    ELSE fehler:=4;        {Variable existiert nicht}
117:   END;
118:
119:   FUNCTION f_wert(formel:string_typ):REAL;
120:   {berechnen des Funktionswertes}
121:   VAR f_name:string_typ;
122:   BEGIN
123:     f_name:='';
124:     WHILE ( Length(formel)>0 ) AND ( formel[1] IN buchstaben) DO BEGIN
125:       f_name:=f_name+formel[1];
126:       formel:=Copy( formel,2,Length(formel) ); END; {formel =argument}
127:     CASE nummer(f_name) OF
128:       0: f_wert:= v_wert( f_name);
129:       1: f_wert:= ArcTan( ergebnis(formel) );
130:       2: f_wert:=    Cos( ergebnis(formel) );
131:       3: f_wert:=    Exp( ergebnis(formel) );
132:       4: f_wert:=    lnn( formel );
133:       5: f_wert:=    Sin( ergebnis(formel) );
134:       6: f_wert:=    Sqr( ergebnis(formel) );
135:       7: f_wert:= wurzel( formel) ;
136:       8: f_wert:= ergebnis( 'PI*(' +formel+ ')/180' );   {Grad->Bogen}
137:       9: f_wert:= ergebnis(formel)*180/Pi;              {Bogen->Grad}
138:      10: f_wert:= Pi; END;                                      {PI}
139:   END;
```

gültigen numerischen Wert, so wird dieser zurückgegeben, andernfalls wird angenommen, daß ein Funktionsaufruf vorliegt.
Der Funktionsaufruf wird in der Funktion **F_Wert** in den Funktionsnamen und das Argument aufgespalten. Anschließend wird der Funktionswert berechnet. Da das Argument wiederum ein zusammengesetzter Ausdruck sein kann, muß er natürlich auch mit der Funktion **Ergebnis** behandelt werden, die hier also indirekt rekursiv aufgerufen wird.
Innerhalb der Funktion F_**Wert** werden die mathematischen Funktionen LN und SQRT durch eigene Funktionen (LNN und WURZEL) berechnet, um Laufzeitfehler bei nichtpositivem bzw. negativem Argument zu vermeiden.
Es können auch beliebige nutzereigene Funktionen – im Beispiel die Umrechnung von Grad- und Bogenmaß – definiert werden. Als Sonderfall ist dabei auch die Definition von Konstanten möglich – z. B. π (PI) –, die einfach als Funktionsaufruf ohne Argument betrachtet werden. Die selbst programmierten mathematischen Funktionen können entweder direkt oder aber wiederum mit der Funktion **Ergebnis** (siehe Berechnung von **Deg** und **Rad** im Programm Val) berechnet werden. Die letzte Variante ist recht elegant, wenn andere nutzereigene Funktionen aufgerufen werden sollen, erfordert aber zusätzlichen Platz auf dem Rekursionsstapel.
Außer der Berechnung von Funktionswerten ist als weitere Spezialfunktion der lesende Zugriff auf Variablen des übergeordneten Programms möglich. Bezeichner, die nicht in der Liste mit den Funktionsnamen enthalten sind, werden als Namen von Variablen aufgefaßt. Da die durch den Nutzer im Quelltext gewählten Namen zur Laufzeit natürlich nicht verfügbar sind, muß der Zusammenhang zwischen Variablenbezeichnung und Variable (genauer der Adresse der Variablen) zur Laufzeit durch die Prozedur **Var_Def** neu festgelegt werden. Damit der Aufruf der Funktion **Ergebnis** ähnlich wie der Aufruf der Standard-Prozedur **Val** gestaltet werden kann, wird sie indirekt über die Prozedur **Valn** aufgerufen. Dieser Prozedur sind dieselben Parameter zu übergeben wie der Prozedur **Val**. Der einzige Unterschied besteht in der Belegung der Variablen **Fehler**. Während bei **Val** in dieser Variablen die Position zurückgegeben wird, an der ein Fehler auftritt, erfolgt bei **Valn** lediglich Rückgabe einer Fehlernummer, die die Art des aufgetretenen Fehlers kennzeichnet. Die Bedeutung der Fehlernummern geht aus Tafel 1 hervor.

Tafel 1 Die Bedeutung der Fehlernummern

Fehler-nummer	Bemerkung
1	Klammerfehler; die Klammern um den bearbeiteten Ausdruck bilden kein zusammengehöriges Paar.
2	Es folgen mehrere Operationszeichen unmittelbar aufeinander.
3	Es wurde versucht, den Logarithmus einer nichtpositiven Zahl zu berechnen.
4	Es existiert keine Variable oder kein Funktionsaufruf mit dem übergebenen Namen.
5	An der ersten Stelle eines Teil-Ausdruckes darf nur ein Vorzeichen (+ oder –) oder eine Zahl stehen.
6	Es wurde versucht die Wurzel einer negativen Zahl zu berechnen.
7	Die zu berechnende Formel ist leer.
8	Division durch 0

```
140:
141:    FUNCTION wert(formel:string_typ):REAL;
142:    VAR f:INTEGER;
143:        w:REAL;
144:    BEGIN
145:      Val(formel,w,f);
146:      IF f = 0 THEN wert:=w                        {Formel besteht aus Zahl}
147:               ELSE wert:=f_wert(formel);      {Formel ist Funktionsaufruf}
148:    END;
149:
150:    FUNCTION ergebnis;                         {wandelt String Real-Zahl}
151:    VAR op1,op2,opz:string_typ;          {1.Operand, 2.Operand, Op.zeichen}
152:        x: REAL;
153:    BEGIN
154:      IF fehler = 0 THEN BEGIN      {Abbruch Rekursion nach erstem Fehler}
155:        formel_aufspalten(formel,op1,opz,op2);
156:        IF op1 = '' THEN
157:          IF ((opz = '-') OR (opz = '+')) AND (op2<>'')
158:            THEN op1:='0'          {Formel zu kompletten Ausdruck kompl.}
159:            ELSE fehler:=5;     {Vorzeichen erwartet oder Operand fehlt}
160:        IF opz = '' THEN
161:          IF ( opz_pos(op1)=0 ) AND       {Ausdruck nicht weiter teilbar}
162:             ( op1[1] <>'(' )                      {und nicht geklammert}
163:            THEN ergebnis:=wert(op1)
164:            ELSE ergebnis:=ergebnis(op1)
165:        ELSE
166:          CASE opz[1] OF
167:            '+': ergebnis:= ergebnis(op1) + ergebnis(op2);
168:            '-': ergebnis:= ergebnis(op1) - ergebnis(op2);
169:            '*': ergebnis:= ergebnis(op1) * ergebnis(op2);
170:            '/': BEGIN
171:                   x:=ergebnis(op2);
172:                   IF x <> 0.0 THEN ergebnis:= ergebnis(op1) / x
173:                                ELSE fehler:=8; END;    {Division durch 0}
174:            '^': ergebnis:=potenz(ergebnis(op1),ergebnis(op2)); END;END;
175:    END;
176:
177: BEGIN {valn}
178:    fehler:=0;
179:    help:='';
180:    FOR n:=1 TO Length(formel) DO         {Leerzeichen weg, Grossbuchst.}
181:      IF formel[n] <> ' ' THEN help:=help + Upcase( formel[n] );
182:    IF help <> '' THEN x:=ergebnis(help)
183:                  ELSE fehler:=7;                           {Formel leer}
184: END;
185:
186: PROCEDURE var_reset;                         {loeschen der Variablenliste}
187: BEGIN
188:    var_akt:=var_anfang;
189:    WHILE var_akt <> NIL DO BEGIN
190:      var_anfang:=var_anfang^.next;
191:      Dispose (var_akt);
192:      var_akt:=var_anfang; END;
193: END;
194:
195: PROCEDURE var_def(name:s10;VAR variable:REAL);
196: {Aufnahme einer Variablen in die Liste}
197: VAR n:INTEGER;
198: BEGIN
199:    FOR n:=1 TO Length(name) DO name[n]:=Upcase(name[n]);
200:    IF var_anfang = NIL THEN BEGIN
201:      New(var_anfang);
202:      var_akt:=var_anfang END
203:    ELSE BEGIN
204:      New(var_akt^.next);
205:      var_akt:=var_akt^.next; END;
206:    var_akt^.name:=name;
207:    var_akt^.zeiger:=Addr(variable);      {bei 8 Bit Ptr(Addr(variable))}
208:    var_akt^.next:=NIL;
209: END;
210:
211: BEGIN {Programm VAL}
212:    var_anfang:=NIL;        {Anfangsinitialisierung, unbedingt notwendig}
213:    var_reset;                             {loescht alte Variablenliste}
214:    var_def('x',x);
215:    var_def('y',y);
216:    WriteLn;
217:    FOR n:=1 TO 9 DO BEGIN        {berechnen Ausdruck mit versch. Werten}
218:      x:=10*n;
219:      valn('rad(x)',y,fehler);
220:      WriteLn(x:10:6,' Grad entspricht im Bogenmass ',y:12:8); END;
221:    WriteLn;
222:    x:=5;                          {Einsetzen Wert in verschiedene Formeln}
223:    valn('x^2',y,fehler);        WriteLn(x:2:2,'^2          =',y:12:6);
224:    valn('x^3',y,fehler);        WriteLn(x:2:2,'^3          =',y:12:6);
225:    valn('Sqrt(x)',y,fehler);   WriteLn('Wurzel aus ',x:2:2,' =',y:12:6);
226:    WriteLn;
227:    REPEAT
228:      WriteLn;
229:      WriteLn('Eigene Formel eingeben, Ergebnis wird in "Y" '+
230:              'abgespeichert, Ende mit ET');
231:      WriteLn;
232:      Read(s);
233:      valn(s,y,fehler);
234:      IF fehler = 0 THEN WriteLn(' = ',y:1:10)
235:                    ELSE WriteLn('   Fehler: ',fehler:3);
236:    UNTIL s='';
237: END.
```

Bild 1

In der vorliegenden Version läuft das Programm unter MS-DOS. Die Anpassung an 8-Bit-Versionen von Turbo-Pascal erfordert eine kleine Änderung in der Prozedur **Var_Def**. Außerdem ist es empfehlenswert, den für den Rekursionsstapel zur Verfügung stehenden Speicherplatz zu vergrößern oder die Länge des Strings für die Formel zu verringern, um den Überlauf des Rekursionsstapels zu vermeiden.

Literatur

/1/ Schneider, M. Taschenrechner-Programmbaustein in Basic. Mikroprozessortechnik, Berlin 2 (1988) 10, S. 313

TERMINE

Internationales Messesymposium „Computersysteme in industriellen Prozessen"

WER? Kammer der Technik e. V., Fachverband Elektrotechnik
WANN? zur Leipziger Frühjahrsmesse '91 am 19./20. März 1991
WO? Leipzig
WAS?
- Hard- und Softwarekomponenten für Prozeßleittechnik und Fertigungsleittechnik
- Vernetzung von Computersystemen und Steuerungsphilosophien für CIM-Projekte
- speicherprogrammierbare Steuerungen
- Computereinsatz im Management

WIE? Einladungen können angefordert werden über:
Hauptgeschäftsstelle der Kammer der Technik e. V., FV Elektrotechnik, Postfach 1315, O-1086 Berlin

Rothstock

2. Berliner UNIX-Forum

WER? IDG World Expo Corporation, Framingham (USA), und UniWare# Computer GmbH, Berlin
WANN? 11./12. Februar 1991
WO? ICC Berlin
WAS?
- offene Systeme
- Datenverarbeitung in öffentlichen Verwaltungen
- Standardisierung
- Office-Systeme
- verteilte Systeme
- objektorientierte Programmiersprachen
- Jahrestagung 1991 der Entwickler- und Anwendergemeinschaft EAG

WIE? Interessenten wenden sich bitte an: UniWare# Computer GmbH, Kantstraße 152, W-1000 Berlin 12; Tel. 31 00 09-2 46

Klauke

3. Fachtagung Computeranimation

WER? Technische Universität „Otto von Guericke"
WANN? 6. und 7. Februar 1991
WO? Technische Universität Magdeburg
WAS?
- Computeranimation für Ausbildung und Training
- Computeranimation und Simulation
- Animationstechnik zur Werbung für Produkte und Verfahren
- Basissoftware für Animation und Simulation

WIE Teilnahmemeldungen bitte an Technische Hochschule Magdeburg, Fakultät für Informatik, PSF 124, O-3010 Magdeburg; Tel. 59 24 46

Lo.

Eine Batchdatei steuert Clipper

Eberhard Klauk, Bensdorf

Moderne Programmiersprachen stehen seit geraumer Zeit als integrierte Entwicklungssysteme wie Turbo-Pascal 5.5, Quick-C 2.0 oder TopSpeed Modula-2 zur Verfügung. Bei der Softwareentwicklung wird hierbei durch interne Editoren und Debugger erheblich Zeit eingespart. Gar nicht zu reden von der Bedienfreundlichkeit.

Das Entwicklungssystem Clipper und der gute alte Assembler müssen nach dem Willen der Entwickler immer noch *von Hand* bedient werden. Hierbei wird nach der konventionellen Methode *Editieren-Kompilieren-Linken-Editieren* ... verfahren, wozu die zugehörigen Tools jedesmal neu aufgerufen werden müssen.
Im folgenden stelle ich Ihnen für die Arbeit mit dem Clipper-Compiler Sommer '87 eine Batchdatei zur Reduzierung der manuellen Eingaben vor, die das gesamte Umfeld von der Quelltexterstellung bis zur lauffähigen EXE-Datei steuert.
Das Programm kann problemlos mit einem Editor erweitert oder optimiert werden. Bis zur Auslieferung von Clipper in integrierter Umgebung sind sicher auch noch Verbesserungen möglich. So könnte beispielsweise bei einem negativen Syntaxtest der Editoraufruf mit der Übergabe der fehlerhaften Zeile erfolgen.
Bild 1 zeigt den Quelltext der Datei CC.BAT. Das dazugehörige Menü wird als CC.MEN abgelegt (Bild 2). Auf diese Weise wird es schneller aufgeblendet als über den Echo-Befehl des Kommandoprozessors. Beim Schreiben dieses Menüs ist darauf zu achten, daß am Anfang eine Leerzeile steht und die letzte Zeile mit <Return> abgeschlossen wird. Das Menü hat einen linken Randabstand von 22 Leerzeichen.
Beim Aufruf ohne Parameter erscheint auf dem Bildschirm eine Hilfsausschrift, die die notwendige Umgebung auflistet. Dabei sind folgende Besonderheiten zu beachten:

- Als Editor sollte der Norton-Editor verwendet werden, da dieser unter anderem das problemlose Editieren mehrerer Quelltexte gleichzeitig oder nacheinander ermöglicht. Eine Zeilennummer kann als Parameter übergeben werden.
- KEY.COM ist eine zur Tastaturabfrage notwendige Assemblerroutine. Außerdem wird damit der Kursor speziell für das Menü gesteuert. Bild 3 zeigt den zugehörigen Assembler-Quelltext, der mit einem beliebigen Editor in die Datei KEY.INP geschrieben werden kann. Die Datei KEY.COM ist am einfachsten mit einem Debugger zu erzeugen.

Dazu eignet sich der zum Lieferumfang der Betriebssysteme PC/MS-DOS gehörende Debugger:
DEBUG < **KEY.INP** <Return>
Zur effektiven Arbeit mit dem Programm sollte ein separates Verzeichnis auf der Festplatte eingerichtet werden, in dem alle Clipper- und Anwenderdateien enthalten sind. Es ist aber auch möglich, Suchpfade für Editor und Linker in der AUTOEXEC.BAT zu deklarieren.
Die Tools zum Anlegen von Reports, Labels, Strukturen oder Indexdateien werden vom Programm CC.BAT nicht genutzt. Da die Report-, Label- sowie Datenbankdateien zu den gängigen Datenbanksystemen dBase III Plus und Foxbase kompatibel sind, können sie auch von diesen erzeugt werden.
Bei umfangreichen Datenbankapplikationen bis hin zur Grafikeinbindung sind natürlich auch die entsprechenden Bibliotheken (Toolboxen) notwendig und müßten in der Link-Zeile von CC.BAT ergänzt werden.
Die Bedienung des Programms ist aus dem Menü sowie der Batchdatei selbst ersichtlich. Über die Option 7 gelangt man in den Kommandoprozessor, um auf DOS-Ebene arbeiten zu können. Diese wird mit Exit wieder verlassen.
Beim Aufruf des Norton-Editors wird zunächst immer der Hauptmodul geladen. Bei modularer Programmierung kann in ein Untermodul nur innerhalb des Editors (<F3>-N-E) gewechselt werden.
Da das Einbinden des Debuggers die EXE-Datei vergrößert, ist nach fehlerfreier Kodierung abschließend mit der Menü-Option 4 zu linken.
Dem Linker sollte man besondere Aufmerksamkeit schenken. Wer nicht unbedingt mit Overlays arbeiten muß, sollte auf den Original-Linker Plink86 verzichten und mit anderen Linkern experimentieren. Bei meinen Tests machte der Linker TLink für den Turbo-Assembler von Borland seinem Namen alle Ehre.
Das Programm wurde auf einem 286er AT

```
@echo off
rem    Batchfile zur Clippersteuerung
rem    Autor: E. Klauk / Brandenburg
:MEN
cls
echo.
echo
echo                  CLIPPER  CONTROLLER  V 3.0
echo
echo.
if "x" == "%1x" goto HELP
echo                          Hauptmodul: %1.prg
type cc.men
key
if errorlevel 57 goto MEN
if errorlevel 56 goto RUN
if errorlevel 55 goto DOS
if errorlevel 54 goto DIR
if errorlevel 53 goto DEB
if errorlevel 52 goto COM
if errorlevel 51 goto TYP
if errorlevel 50 goto SYN
if errorlevel 49 goto EDI
if errorlevel 48 goto END
goto MEN
:DIR
cls
dir *.prg/p
pause
goto MEN
:DOS
cls
command
goto MEN
:EDI
ned %1.prg
goto MEN
:COM
cls
if exist pro.txt del pro.txt >nul
if exist %1.exe del %1.exe >nul
if exist %1.obj del %1.obj >nul
clipper %1 -l >pro.txt
if not errorlevel 1 goto LIN
goto TYP
:LIN
echo.
echo.
tlink %1,,nul,extend+clipper
goto ANZ

:DEB
if not exist %1.obj goto MEN
cls
tlink/v %1+debug,,nul,extend+clipper
:ANZ
dir %1.exe
pause
goto MEN
:SYN
cls
if exist pro.txt del pro.txt >nul
clipper %1.prg -s >pro.txt
:TYP
cls
type pro.txt
pause
goto MEN
:RUN
if exist %1.exe %1
goto MEN
:HELP
echo Entwicklungssystem zum Entwerfen und Testen von Datenbankapplikationen
echo.
echo Controller-Aufruf:        [LW:]CC Hauptmodul
echo.
echo Erforderliche       CLIPPER.EXE    (Compiler)
echo     Controller-     CLIPPER.LIB    (Laufzeitbibliothek)
echo       umgebung:      EXTEND.LIB    (Erweiterungsmodul)
echo                       DEBUG.OBJ    (Debuggermodul)
echo                       TLINK.EXE    (Programmverbinder)
echo                         KEY.COM    (Controllerhilfe)
echo                         NED.COM    (Quelltexteditor)
echo                          CC.MEN    (Controllermenü)
:END
echo.
```

***Bild 1** Die Datei CC.BAT steuert Clipper.*

```
Editieren.......................1
Syntaxprüfung...................2
Letztes Protokoll...............3
Compilieren/Linken..............4
Einbinden Debugger..............5
Quellenverzeichnis..............6
Kommandoprozessor...............7
Start Hauptmodul................8
E n d e.........................0

Gib eine Controlleroption ein : :
```

***Bild 2** Das Menü steht in der Datei CC.MEN; der linke Randabstand beträgt 22 Leerzeichen.*

```
n key.com
r cx
1a
a
mov ah,03
mov bh,00
int 10
sub dh,01
add dl,35
mov ah,02
int 10
mov ax,0c01
int 21
mov ah,4c
int 21

w
q
```

***Bild 3** Der Assembler-Quelltext für die Datei KEY.INP. Die Leerzeile nach int 21 muß sein!*

mit Festplatte getestet. Bei Maschinen mit ausschließlichem Diskettenbetrieb sollten CC.BAT, NED.COM und KEY.COM in der RAM-Disk laufen, um die Diskettenzugriffe zu reduzieren. Die entsprechenden Kommandozeile in CC.BAT sind in diesem Fall um die Laufwerksangaben entsprechend der Dateiverteilung zu erweitern. Der Aufruf erfolgt dann vom aktuellen Arbeitslaufwerk.

Zweites Floppylaufwerk mit 80 Spuren

Wolfgang Schreiber, Chemnitz

Mit dem 80spurigen Floppylaufwerk K 5601 von Robotron Ascota (das für rund 200 Mark im Fachhandel erhältlich ist) ergeben sich interessante Anwendungen im PC-Bereich. Der folgende Beitrag gibt deshalb Tips für die Nachrüstung von XTs und ATs.

PC/XT

Das Hausformat des PC/XT (360 KByte) und die dafür eingesetzten 40spurigen Floppylaufwerke stellen eine Begrenzung des Datenaustausches zu anderen Rechnern bzw. Betriebssystemen dar. Ein Einbau des K 5601 als Zweitlaufwerk ermöglicht eine Erhöhung der Datendichte auf DD-Disketten und die Übernahme bzw. Übergabe von Daten von/zu allen DDR-PCs (z. B. EC 1834: 720 KByte, PC 1715: 780 KByte).

Der Einsatz eines 80spurigen Laufwerkes als Laufwerk A: ist zwar prinzipiell möglich, aber wenig sinnvoll, da viele Installationsprogramme, neue Betriebssystemversionen usw. ein Laufwerk A: mit 360 KByte voraussetzen.

Hardware

① Das Hosenträgerkabel vom Controller zu den Laufwerken muß 2 Laufwerkssteckverbinder haben.

② Die Systemboard-Switches sind wie folgt einzustellen: 7 = off, 8 = on.

③ Auf dem Laufwerk K 5601 sind die Jumper DS1, XT, FG und E2 zu setzen.

④ Der Abschlußwiderstand ist auf dem 1. Laufwerk nach dem Controller zu entfernen.

Nutzungsmöglichkeiten

Das Laufwerk B: kann 720 KByte ohne softwaremäßige Einstellungen lesen. Die Verarbeitung von 360 KByte ist nicht möglich. Falls das Programmpaket SuperCopy zur Verfügung steht, installiert man ein 96-tpi-Laufwerk (-r22). Damit wird die Verarbeitung (Lesen, Schreiben, Formatieren) der verschiedenen DOS-Formate über MDRIVE sowie von CP/M-Formaten über CDRIVE auf dem entsprechenden logischen Laufwerk möglich.

Eine Nutzung über Driver.sys ist möglich; der Vorteil ist: Das logische Laufwerk gestattet die Anwendung von MS-DOS-Befehlen wie FORMAT, DISKCOPY usw. Hier ein Beispiel für die Einstellung der Config.sys:

device = driver.sys /d:1 /t:80 /s:9 /h:2 /f:2

Das Laufwerk B: wird über IOCTL umgestellt; der Vorteil ist: Auch die Formatierung von 720 KByte wird möglich. Dafür wäre das Programm b_set_xt.com in Bild 1 in der Autoexec.bat aufzurufen.

Sollte das Laufwerk nicht fehlerfrei arbeiten oder starke Geräusche verursachen, so ist DISK_BASE (Pointer Int 1EH) zu analysieren bzw. zu modifizieren. /1/ erläutert DISK_BASE näher.

PC/AT

Der AT wird standardmäßig mit nur einem 1,2-MByte-Laufwerk ausgerüstet. In vielen Anwendungen erscheint ein zweites Laufwerk als wünschenswert.

Hardware

① Das Hosenträgerkabel vom Controller zum Laufwerk muß ebenfalls 2 Laufwerkssteckverbinder haben.

② Auf dem Laufwerk K 5601 sind die Jumper DS1, XT, FG und E0 zu setzen.

③ Der Abschlußwiderstand auf dem 1. Laufwerk nach dem Controller ist zu entfernen.

Nutzungsmöglichkeiten

Die Nutzung als 720-KByte-Laufwerk ist dann möglich, wenn das ROM-BIOS eine solche Einstellung unterstützt (Setup: 720 KByte). Für die Verwendung von 360-KByte-Disketten gehen Sie wie folgt vor: Um das 96-tpi-Laufwerk auf 48 tpi umzuschalten, ist der Widerstand R 19 unterhalb der Pins 4 und 6 des Schaltkreises 6460A0 zu entfernen (Setup: 360 KByte). Die Disketten sind voll kompatibel zu 96-tpi-Laufwerken (tpi – tracks per inch). Die Austauschbarkeit zu „echten" 48-tpi-Laufwerken unterliegt den in /2/ genannten Einschränkungen.

Es ist zu beachten, daß das Laufwerk K 5601 hardwaremäßig keinen Diskettenwechsel erkennt. Dieser Sachverhalt ist unbedingt dem Betriebssystem über IOCTL bekannt zu machen, da sonst bei Diskettenwechsel kein gültiges Directory zur Verfügung steht. Dafür könnte das Programm b_set_at.com in Bild 2 in der Autoexec.bat aufgerufen werden.

Die Anwendung des Programmpaketes SuperCopy ist mit Problemen behaftet. Der Schalter -r22 unterstützt zwar Diskettenwechsel, schaltet aber nach einem Zugriff auf A: den HD/FD-Controller nicht auf die erforderliche Übertragungsrate zurück. Deshalb rate ich von einer Anwendung von SuperCopy auf dem K 5601 im AT ab.

```
debug
f cs:100 200 00
a 100
mov ax,440d
mov bl,2
mov ch,8
mov cl,40
mov dx,112
int 21
mov ah,4c
int 21
db 00 02 00 00 50 00 00 00
db 02 02 01 00 02 70 00 a0
db 05 f9 03 00 09 00 02 00
^c
n b_set_xt.com
r cx
100
w
q
```

Bild 1 DOS-Programm für XTs

```
debug
f cs:100 200 00
a 100
mov cl,60
call 130
and byte ptr [152],fd
mov cl,40
call 130
mov ah,4c
int 21
^c
a 130
mov ax,440d
mov bl,2
mov ch,8
mov dx,150
int 21
ret
^c
n b_set_at.com
r cx
100
w
q
```

Bild 2 DOS-Programm für ATs

Tafel 1 Verwendete Hard- und Software

PC	Karten	BIOS	DOS
XT	Multi-I/O	AMI-BIOS 10/03/86	MS-DOS 3.30
AT	HD-/FD-Controller -WD1003V-MM2 -WD1006V-SR2 -GR87132	-DTK 01/25/89 -IBM 11/15/85	MS-DOS 3.30 PC-DOS 3.30

Literatur

/1/ Robotron-Büromaschinenwerk: Anleitung für den Systemprogrammierer DCP 3.20. Sömmerda 1987
/2/ IBM Deutschland GmbH: DOS 3.30 Bedienerhandbuch. BRD 1987.
/3/ Smode, D.: Das große MS-DOS-Profi-Arbeitsbuch. Francis-Verlag, München 1987
/4/ Robotron-Buchungsmaschinenwerk: Minifolienspeicher robotron K 5601. Karl-Marx-Stadt 1987

✉ KONTAKT ☎ Forschungszentrum des Werkzeugmaschinenbaues, Abteilung Rechentechnik, Karl-Marx-Allee 4, O-9010 Chemnitz; Tel. 5 99-4 90

TERMINE

2. Dresdner Informatiktage

WER? Technische Universität Dresden, Informatik-Zentrum
WANN? 20. und 21. Februar 1991
WO? Dresden
WAS?
- Computer Aided Design
- Computer Aided Manufacturing
- Theorien und Methoden
- Methoden der Künstlichen Intelligenz
- Aus- und Weiterbildung

WIE? Interessenten wenden sich bitte an: Technische Universität Dresden, Informatik-Zentrum – SWB 2. Dresdner Informatiktage, Mommsenstraße 13, O-8027 Dresden; Tel. 4 57 53 90 (Prof. Dr. sc. H. Stahn), 4 57 53 58 (Tagungsbüro)

Microsoft Quick-Pascal 1.0

Turbo versus Quick

Uwe Schulze, Berlin

Lange Zeit wurde Pascal im Hause Microsoft recht stiefmütterlich behandelt; man setzte auf C und -Basic. Die außerordentlichen Erfolge von Turbo-Pascal zwangen nachzuziehen. MP stellt Ihnen das neue Quick-Pascal vor.

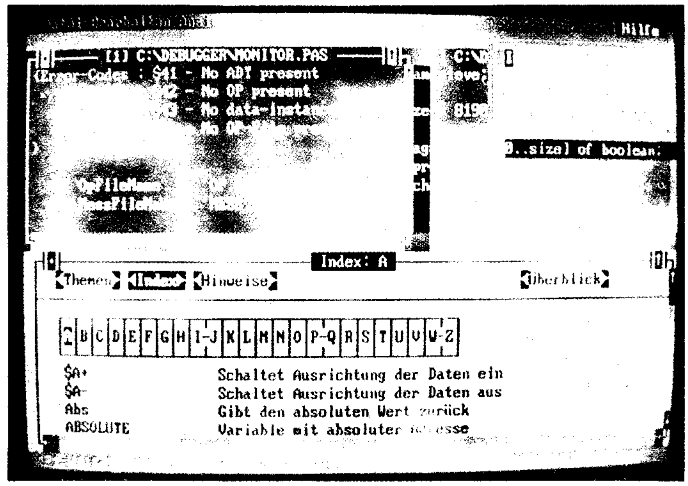


Bild 2

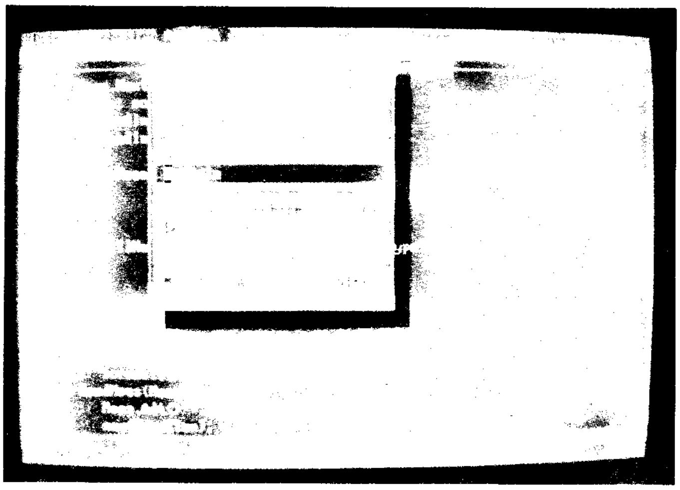


Bild 1

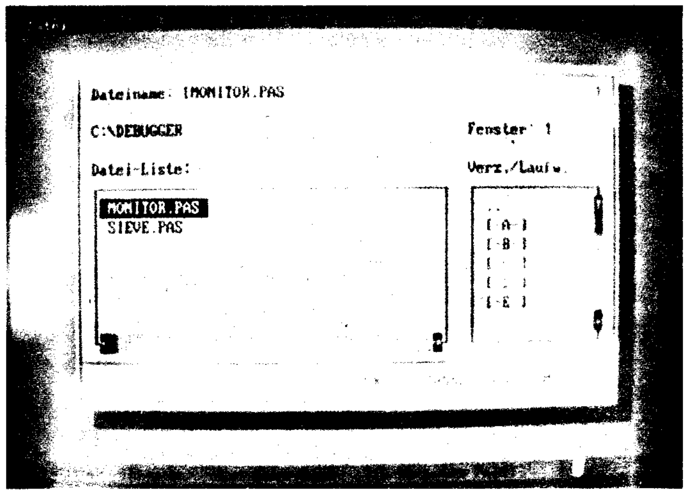


Bild 3

Die Entwicklungsumgebung

Die integrierte Nutzeroberfläche präsentierte sich mit gleichem Aussehen wie die anderen Quick-Sprachen (C, Basic) und das auch in einer deutschen Version. Damit sind nicht nur die Hilfstexte gemeint (die gibt es bei Borland auch), sondern das gesamte Menüsystem. Ob mit der deutschen Übersetzung Kacheln aber mehr anzufangen ist, als mit der Originalausschrift mag jeder für sich selbst entscheiden (Bild 1). Die Oberfläche entspricht IBMs SAA-Spezifikation (Systems Application Architecture) und ähnelt damit Anwendungen unter MS-Windows, der DOS 4.0-Shell oder dem Präsentation-Manager von OS/2.

Typisch dafür sind die mit dem Kursor bzw. dem Leuchtbalken mitwandernden Blöcke an den Seiten der Fenster, die beispielsweise den Stand des Kursors in einer Datei oder in einem Verzeichnis anzeigen, und die Menü-Boxen (siehe Bild 3). Die Änderung der Standardeinstellungen der Fenster (z. B. Farben, Größe und Lage) kann direkt in der Entwicklungsumgebung erfolgen; ein separates Installatinsprogramm ist nicht nötig. Der **Editor** bietet wesentlich mehr, als der bekannte Turbo-Editor. Es handelt sich um einen *Multi-Window-Editor*, mit dem beliebig viele Quelltexte, Hilfstexte und Debug-Informationen in verschiedenen Fenstern überlappend oder nicht-überlappend dargestellt werden können (siehe Bild 2). Die Maus wird unterstützt, was bei der Arbeit in mehreren Fenstern aber auch nötig ist. Die meisten Menüpunkte sind zusätzlich über bestimmte Tasten(-kombinationen), sogenannte Shortcuts oder Hotkeys, zu erreichen, doch unterscheiden sich alle von denen in Turbo-Pascal: (<Alt>-<F4> statt <Alt>-X für Beenden, <F5> statt <Ctrl>-<F9> für den Programmstart usw. Übersichtlich und sehr umfangreich ist die kontextsensitive Online-Hilfe mit Beispielprogrammen, die – wie in Turbo-Pascal auch – in die Anwendung hinein kopiert werden können. Allerdings wird nicht so konsequent mit Querverweisen gearbeitet, wie bei Turbo.

Unter den mitgelieferten Demo-Programmen befindet sich auch ein Sortierprogramm, das sechs verschiedene Sortieralgorithmen grafisch darstellt und damit verständlich macht. Wünschenswert wären mehr Anregungen zur objektorientierten Programmierung. Hier wird der Programmierer mit einem kleinen Beispiel ziemlich allein gelassen.

Die Informationen über die System-Units sind allerdings *nur* über die Hilfstexte zu erfahren, während Borland sie als Textdatei ausliefert. Etwas ganz besonderes hat man sich mit dem sogenannten Color-Coding ausgedacht: Pascal-Schlüsselworte, Zeichenketten und Kommentare werden verschiedenfarbig dargestellt, so daß vergessene Anführungszeichen oder Kommentarklammern sofort auffallen. Die Farbeinstellung kann durch jeden Anwender leicht nach seinen Wünschen geändert werden. Neben der integrierten Entwicklungsoberfläche wird – wie bei Turbo-Pascal auch – ein separater Kommandozeilencompiler (QPL.EXE) ausgeliefert.

Der Compiler

Quick-Pascal hält sich an den von Turbo-Pascal 5.0 gesetzten Sprachstandard, und so wird das nicht eben kleine Programm BGIDEMO ohne Probleme übersetzt und abgearbeitet (Tafel 1). Gleiches gilt für die gängigen Toolboxen – allerdings stößt man schneller an die Speichergrenzen, da durchgängig großzügiger mit dem Speicherplatz umgegangen wird als bei Borlands Compiler. Das ist auch deutlich an der Größe der System-Units ablesbar (Tafel 2). Diese sind nicht – wie bei Turbo-Pascal – in einer Bibliothek zusammengefaßt (dort TURBO.TPL) und können damit auch nicht im Speicher gehalten werden. Alle vorkompilierten Units tragen die Erweiterungen QPU (Quick Pascal Unit). Die Namen der System-Units sind identisch – nur die Quick-Grafik-Bibliothek trägt den Namen MSGRAPH. Zusätzlich existiert aber noch die Kompatibilitäts-Unit GRAPH, so daß auch BGIDEMO ohne Änderungen an den Uses-Anweisungen übersetzt werden konnte. Von Quick-Pascal nicht unterstützt wird die Overlaytechnik, die ja auch bei Turbo-Pascal 4.0 aus dem Programm genommen wurde, seit der Version 5.0 aber wieder existiert und sich in 5.5 weiter verbessert findet. Deshalb also keine Unit OVERLAY in Quick-Pascal.

Als Schnittstelle zu C-Programmen – und abweichend von Sprachstandard – kennt Quick-Pascal auch nullterminierende Strings, die durch das reservierte Wort CSTRING vereinbart werden können.

So erzeugt

CONST str: CSTRING = '4711';

eine Zeichenkette, die mit einer binären Null abschließt.

Der Compiler beinhaltet ferner eine

Tafel 1 Die Kompilierung des Programms BGIDEMO mit den Kommandozeilen-Compilern QPL bzw. TPC

	Quick-Pascal	Turbo-Pascal
Codegröße (Byte)	89488	41920
Compilezeit (s)	17.9	14.2

Tafel 2 Die System-Units

Quick-Pascal		Turbo-Pascal	
SYSTEM.QPU	44976 Byte	SYSTEM.TPU	30384 Byte
CRT.QPU	5152 Byte	CRT.TPU	3968 Byte
DOS.QPU	9104 Byte	DOS.TPU	5856 Byte
PRINTER.QPU	304 Byte	PRINTER.TPU	432 Byte
GRAPH.QPU	37644 Byte	GRAPH.TPU	31584 Byte
MSGRAPH.QPU	72368 Byte	---	
---		OVERLAY.TPU	3712 Byte

Option zur Erzeugung von 286er Code, die Turbo-Pascal nicht kennt. Der integrierte Debugger entspricht etwa dem von Turbo-Pascal.
Für die Programmbearbeitung können verschiedene Geschwindigkeiten eingestellt werden, um das Programm dann wie von Geisterhand zu tragen und ggf. anzuhalten. Während Borland aber noch einen exzellenten Stand-Alone-Debugger (einschließlich objektorientierter Funktion) liefert, arbeitet Microsofts Produkt Codeview noch nicht mit Quick-Pascal zusammen.
Mit Quick-Pascal erzeugte Programme sind nicht nur wesentlich größer als deren Turbo-Pendants (was auch eine längere Übersetzungszeit bedingt), sie sind auch beträchtlich langsamer. Insbesondere die Arithmetik und die Grafik schneiden schlechter ab. Leider macht der Quick-Pascal-Compiler auch das einzige echte Manko der Borland-Sprachen – keine Erzeugung Microsoft-kompatiblen Objektkodes zum Linken mit fremdsprachigen Modulen – nicht wett.

Auf Objekte orientiert

Die objektorientierten Erweiterungen entsprechen ziemlich genau denen von Turbo-Pascal 5.5; auch die Grenzen sind die gleichen: Ein Überladen von Operatoren und die Verwendung einer Methode mit verschiedenen Parametern sind nicht möglich. Drei Schlüsselworte sind für die objektorientierte Programmierung hinzugekommen:
Object, Override und **Inherited.**
Während Turbo-Pascal aus Gründen der Performance standardmäßig statische Methoden generiert und dynamische Methoden explizit deklariert werden müssen; kennt Quick-Pascal nur danamische Methoden. Beim Überladen von Methoden muß dies mit Override bekannt gegeben werden.
Ein Beispiel der Klasse eines grafischen Punktes aus dem Turbo-Pascal-Handbuch 5.5, S. 37

```
type
  point = object (location)
  visible: boolean;
  constructor Init(InitX, InitY: integer);
  procedure Show; virtual;
  porcedure Hide; virtual;
  function IsVisible: boolean;
  procedure MoveTo(NewX,NewY:
  integer);
  end;
```

hätte in Quick-Pascal folgendes Aussehen:

```
type
  point = object (location)
  visible: boolean;
  procedure Init(InitX, InitY: integer);
  procedure Show;
  procedure Hide;
  function IsVisible: boolean;
  procedure MoveTo(NewX,NewY:
  integer);
  end;
```

Um in einem weiteren Objekt die Prozedur Show zu überladen, wäre folgende Anweisung nötig:

```
new_point = object(point)
  procedure Show; override;
  end;
```

Wie am Beispiel zu sehen, kennt Quick-Pascal die Begriffe des Konstruktors und Destruktors nicht. In C++ (das als Vorbild steht) sind ebenfalls keine Schlüsselwörter dafür vorgesehen, wohl aber Konstruktor- und Destruktor-Funktionen. Diese werden dort über eine Funktion, die den Klassennamen trägt (für die Konstruktor-Funktion), bzw. den Klassennamen mit einem vorangestellten Wiederholungszeichen (~) aufgerufen. In Quick-Pascal erfolgt die Initialisierung automatisch, so daß ein expliziter Konstruktor entfällt. Soll innerhalb einer Klasse eine gleichnamige Methode aus einer übergeordneten Klasse (also eine überladene Methode) ausgeführt werden, so steht dafür das Schlüsselwort Inherited zur Verfügung. Der Aufruf der Prozedur Show aus der Klasse Point innerhalb von Methoden der Klasse New_point würde mit
Inherited self.Show;
erreicht werden. Die Qualifizierung von Methoden der eigenen Klasse erfolgt wie bei Turbo-Pascal mit Self, allerdings ist die Angabe hier nicht optinal.

Wegbereiter der Informatik

ALAN MATHISON TURING

∗ 1912 London
† 1954 Wilmslow (Cheshire)

Der englische Mathematiker A. Turing galt schon in seiner Kindheit als außerordentlich intelligent und zeigte dabei einen deutlichen Hang zum Außenseiter. Mit 18 Jahren trat er in das berühmte Kings-College in Cambridge ein. Sechs Jahre später – also 1936 – setzte er seine Studien am Graduate College in Princeton fort, wo u. a. Albert Einstein sowie die Mathematiker John v. Neumann, Hermann Weyl und Richard Cuorant lehrten. Hier promovierte Turing mit der Dissertation „System of Logic based on Ordinals". Ein Angebot, Assistent bei J. v. Neumann zu werden, lehnte er ab und kehrte 1939 wieder nach England zurück, um sich bis Kriegsende der Nachrichtenabteilung des Foreign Office als Kryptologe zur Verfügung zu stellen.
Nach dem Krieg schloß er sich für zwei Jahre einem Team des National Physical Laboratory in Teddington an, das den ersten englischen Digitalrechner – die ACE (Automatic Computing Engine) – entwikkelte. Die ACE war eine Variante der amerikanischen ENIAC und stellte erstmals sowohl Zahlen als auch Befehle gleichermaßen als Binärmuster dar.
Im Jahre 1948 nahm Turing eine Dozentur an der Universität Manchester an; dort wurde ebenfalls ein Digitalrechner konstruiert, für den er 1950 eine Programmieranleitung verfaßte. Auf Vorschlag von A. Newmann und Bertrand Russell wurde er 1952 zum Mitglied der Royal Society gewählt.
In Fachkreisen besonders bekannt geworden ist A. Turing durch seinen 1937 in den Proceedings der Londoner Mathematischen Gesellschaft veröffentlichten Beitrag zum Hilbertschen Entscheidungsproblem. Der damals in Göttingen lebende bedeutende Mathematiker David Hilbert (1862–1943) hatte in einem berühmt gewordenen Vortrag auf der 1. Mathematikertagung 1900 in Paris zum Wesen der Mathematik 23 Probleme formuliert, die bis heute das Interesse der mathematischen Fachwelt besitzen. Eines davon ist das Entscheidungsproblem; Turing bewies, daß es – im Gegensatz zu Hilberts Vermutung – generell unlösbar ist. Das bedeutet, daß es zum Beispiel in der mathematischen Logik Aussagenkalküle gibt, für die mit keinem Algorithmus nach endlich vielen Schritten entschieden werden kann, ob sie allgemeingültig sind oder nicht.
Für seinen Beweis erdachte Turing einen extrem einfachen, idealisierten Automaten, die sogenannte „Turing-Maschine". Mit dieser fiktiven Maschine definierte er den Begriff der Berechenbarkeit bzw. des Algorithmus und konstruierte dann eine Klasse von mathematischen Problemen, die durch einen algorithmischen Prozeß nicht gelöst werden können. Er zeigte die Äquivalenz von Algorithmus und Programm auf und wies damit nach, daß es Computerprogramme gibt, die sich mit keinem automatischen Verfahren auf ihre Korrektheit überprüfen lassen. Das Turingmaschinenmodell wurde so zur Grundlage der Berechnungstheorie und der Komplexitätstheorie – beides Zweiggebiete der theoretischen Informatik.
Sehr viele biographische Angaben über A. Turing, besonders aus der Zeit des Krieges, verdankt die Nachwelt seiner engsten Mitarbeiterin, die seit 1939 ein Institutstagebuch führte, das aus Geheimhaltungsgründen nicht vor 1974 an die Öffentlichkeit gegeben werden durfte. Daraus wird ersichtlich, daß Turing den Geheimcode der von der deutschen Wehrmacht benutzten Chiffriermaschine „Enigma" (abgeleitet vom griechischem Wort ainigma = Rätsel) entschlüsselt hat. Diese Leistung wurde für England kriegsentscheidend; denn der englischen Heeresleitung wurde es dadurch ermöglicht, die Funksprüche der deutschen Wehrmacht seit 1940 abzuhören. Für die Decodierung der Enigma (nebenstehende Abb.), die 310 verschiedene Codes zuließ, stand Turing ein Exemplar dieser Maschine zur Verfügung, das polnische Agenten 1930 aus einer Berliner Fabrik entwenden konnten. Ausführlicheres darüber kann man zum Beispiel bei Rolf Hochhuth nachlesen (Rowohlt 1987), der das erwähnte Tagebuch literarisch verarbeitet hat.

Dr. Klaus Biener

Einführung in MS-DOS

Teil 5

Wolfram Schulze, Uwe Schulze, Berlin

Der Norton Commander

Nicht alles ist so schwer, wie in den bisherigen vier Folgen beschrieben – es gibt ja Dienstprogramme. Heute vorgestellt: Der Norton Commander.

Dienstprogramme und Nutzeroberflächen

In den ersten vier Folgen unseres DOS-Kurses wurden die Kommandos in sehr konservativer Weise behandelt. Nun hebt sich aber gerade DOS von anderen Betriebssystemen dadurch ab, daß eine Reihe komfortabler Bedienoberflächen, Dienstprogramme, Fullscreen-Editoren und Tools aller Couleur verfügbar sind. Zumindest drei Argumente sprechen aber für eine ausführliche Behandlung der DOS-Kommandooberfläche: Erstens läßt sich nur so ein echtes Verständnis für die innere Struktur von DOS vermitteln. Zweitens kann man die Befehle in Batch-Dateien zusammenfassen oder auch in Programmen verwenden (indem der Kommandointerpreter ausgeführt wird). Und drittens: Was tun Sie, wenn einmal ein Dienstprogramm zur Hand ist?

Nun war von einfacher Bedienung (die die weite Verbreitung von MS-DOS gefördert haben soll) an mancher Stelle der vorangegangenen Kurs-Folgen nichts zu merken – denken Sie nur an die Batch-Dateien. Dieser Mangel an Bedienkomfort war wohl auch den Betriebssystem-Herstellern bekannt. Sie lieferten nachträglich grafische Nutzeroberflächen (Windows, GEM), die insbesondere dem Einsteiger den Kommandodschungel ersparen sollen. Bei den neuen Betriebssystemen wird der Befehlsinterpreter von einer menügeführten Nutzeroberfläche (Shell) abgelöst (DOS-Shell bei MS-DOS 4.0, Presentation Manager bei OS/2). Ihr Vorteil liegt in einem einheitlichen äußeren Erscheinungsbild, an das sich auch immer mehr andere Software anlehnt und das IBMs SAA (Systems Application Architecture) entspricht. In jedem Falle werden aber ein grafikfähiges Gerät und eine Maus benötigt.

Neben den erwähnten Shells existieren noch eine Reihe von Dienstprogrammen (Pctools, Qdos, Xtree), die anstelle der internen und externen Systembefehle eingesetzt werden und die statt langer Kommandofolgen nur weniger Handgriffe bedürfen. Auf eine Gegenüberstellung dieser Programme wurde hier verzichtet – jeder mag verwenden, was er am besten beherrscht und was den Anforderungen entgegenkommt. Statt dessen wird der Norton Commander – als ein typischer Vertreter – genauer vorgestellt, da er weite Verbreitung erlangt hat und viele nützliche Eigenschaften der oben genannten Bedienoberflächen und Dienstprogramme in sich vereint.

Die Bedienung des Norton Commanders

Der Start des Norton Commanders kann auf zwei Wegen erfolgen. Ruft man Nc.exe auf, so wird das gesamte Programm resident in den Speicher gelegt, und das kostet immerhin 140 KByte Hauptspeicher. Da sich viele Anwendungen darauf gar nicht erst einlassen (in Turbo-Pascal kann man nur noch wenige Zeilen große Programme übersetzen), wird man sich zumeist für die zweite Variante entscheiden. Dazu installiert man mit Ncsmall.-exe ein kleines residentes Programm, das nur etwa 12 KByte Speicher permanent belegt und den Rest des Norton Commanders nachlädt – diesen Speicherplatz aber freigibt, wenn ein anderes Programm gestartet wird. Eine solche Verfahrensweise ähnelt der bei Command.com, wo auch ein residenter und ein transienter (nachzuladener) Teil unterschieden werden. Der Nachteil liegt auf der Hand: Das Nachladen kostet Zeit. Hier kann man sich wiederum helfen, indem man Nc.exe auf eine RAM-Disk kopiert.

Ein Blick auf die Festplatte (oder Diskette) zeigt Ihnen, daß noch weitere Dateien zum Norton Commander gehören. Sie dienen insbesondere der Anpassung an die individuellen Wünsche des Nutzers. Im einzelnen können folgende Files vorhanden sein:

Nc.ini
Nc.ext (Textdatei)
Nc.mnu (Textdatei)
Treeinfo.ncd

Später werden wir sehen, wie diese Dateien modifiziert (oder, falls noch nicht vorhanden, angelegt) werden. In die Textdateien kann aber schon jetzt mit einem Editor oder mit Type ein Blick geworfen werden. Die ini-Datei (ini für Initialisation) enthält Angaben zur Konfiguration, betrifft also das Outfit, und das kann recht unterschiedlich sein. In Nc.ext (ext für Extension) sind Kommandos eingetragen, die in Abhängigkeit von der Dateierweiterung ausgeführt werden, Nc.mnu (mnu für Menü) beinhaltet nutzerdefinierte Menüs und in Treeinfo.ncd merkt sich Norton den Verzeichnisbaum, was insbesondere bei großen Festplatten zu erheblicher Zeitersparnis führt. Im folgenden wird auf die weit verbreitete Version 2.0 Bezug genommen, inzwischen wird bereits die Folgeversion 3.0 geliefert. Dort trägt das Hauptprogramm den Namen NCMAIN.EXE, und der residente Teil heißt NC.EXE, so daß ein NCSMALL entfällt.

Beim Aufruf erscheint das Programm wie in Bild 1 – in Abhängigkeit von den Eintragungen in Nc.ini, aber auch ein wenig anders. Zwei Verzeichnisfenster bedecken fast den gesamten Bildschirm. Sie werden Panels (engl., = Tafel, rechteckiges Format) genannt und enthalten die Datei- und Verzeichniseinträge. In einem Fenster befindet sich ein Balkencursor, der mit den Pfeiltasten auf und ab bewegt werden kann. Stellt man ihn auf den obersten oder untersten Eintrag, so wird automatisch gescrollt (Kunstwort aus screen und rolling), also die Einträge werden um eine Position nach unten oder oben verschoben. So sind immer alle Dateieinträge zu erreichen. Weiterhin ist das Blättern mit <P-gUp> bzw. <Bild ↑> und <PgDn> bzw. <Bild ↓> sowie das direkte Anwählen des ersten Eintrages mit <Home> bzw. <Pos 1> und des letzten mit <End> bzw. <Ende> möglich.

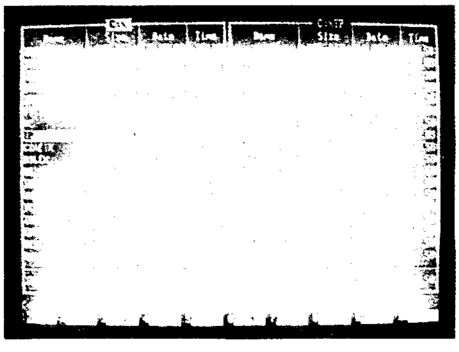


Bild 1 In dieser Einstellung belegt der Norton Commander den gesamten Bildschirm

Ein großer Vorteil des Norton Commanders besteht darin, daß auch die traditionelle Kommandozeile jederzeit zur Verfügung steht. In Bild 1 sehen Sie auf der zweiten Zeile von unten den DOS-Prompt, und dahinter steht der blinkende Cursor. Es ist also kein Problem, einmal ein **dir** einzugeben. Dann passiert genau das, was man erwartet – für die Zeit der Ausführung verschwinden aber die Verzeichnisfenster des Commanders. Und so hält er es immer, wenn andere Programme gestartet werden. Den DIR-Befehl wird man wahrscheinlich nicht mehr verwenden, aber probieren Sie doch einmal einen Laufwerkwechsel. Dazu ist einfach **a:** einzugeben (sofern auch eine Diskette eingelegt ist). Sie werden feststellen, daß anschließend auch die Datei von Laufwerk A: im aktuellen Panel angezeigt werden. DOS und der Norton Commander arbeiten also Hand in Hand. Zum Wechseln des Laufwerkes kann man aber auch eine Tastenkombination verwenden, nämlich <Alt>-<F1> für das linke und <Alt>-<F2> für das rechte Fenster. Es erscheint dann ein kleines Menü, aus dem das neue Laufwerk ausgewählt werden kann. Das geht auch nicht schneller als in der Kommandozeile, hat aber den Vorteil, daß alle vorhandenen logischen Laufwerke (also auch mit Supercopy eingerichtete oder im Netzwerk angemeldete) aufgelistet werden.

In den beiden Panels sind nicht nur unterschiedliche Laufwerke einstellbar, sondern auch unterschiedliche Verzeichnisse eines Laufwerks. Traditionell kann dazu **cd** für Change Directory verwendet werden, einfacher ist es aber, den Balkencursor auf den entsprechenden Directory-Eintrag zu positionieren und <Enter> zu drücken. Verzeichnisse sind im Gegensatz zu Dateien an den Großbuchstaben zu erkennen; außerdem sind sie mit **SUB-DIR** gekennzeichnet. Wie gehabt stehen für das übergeordnete Verzeichnis zwei Punkte. Im linken Panel von Bild 1 gibt es kein übergeordnetes Verzeichnis; wir befinden uns im Hauptverzeichnis, was auch an der Pfadangabe in der obersten Zeile zu sehen ist. Der Prompt der Kommandozeile zeigt das Verzeichnis des aktuellen Fensters (zu erkennen am Balkencursor) an. Wechseln Sie doch einmal mit <Ctrl>-<I>

oder der Tabulatortaste in das andere Verzeichnisfenster und achten Sie auf den Prompt. Anschließend sollten Sie versuchen, ein anderes Directory einzustellen (cd bleibt Ihnen dafür immer), indem Sie den Balkencursor auf einen Directory-Eintrag setzen und mit <Enter> abschließen.
Die Directories stehen stets vor den Dateien im Fenster, so daß man mit <Home> bzw. <Pos1> schnell nach oben blättern kann. Da Verzeichnisoperationen sehr häufig benutzt werden, gibt es noch eine Reihe von Tastenkombinationen (Shortcuts) zum schnellen Wechsel. So gelangt man mit <Ctrl>-<PgUp> bzw. <Strg> <Bild↑> von jeder Stelle im Panel in das übergeordnete Verzeichnis, und mit <Ctrl>-<\> bzw. <Strg>-<\> wechselt man in das Hauptverzeichnis (Root) (\ bezeichnet dabei die Taste, nicht das Zeichen).
Es gibt noch eine weitere Möglichkeit, schnell zu einer bestimmten Datei im aktuellen Verzeichnis zu gelangen. Dazu tippen Sie <Alt> und den Anfangsbuchstaben ein. Der Balkenkursor springt auf den ersten Eintrag dieses Buchstabens. Die Angabe der folgenden Buchstaben (sofern Sie das wollen) kann auch ohne <Alt> erfolgen. Ein kleines Suchfenster am unteren Panelrand hält Sie darüber auf dem laufenden, ob überhaupt Dateinamen mit dieser Buchstabenfolge vorhanden sind.
Nun soll die Bedienung einer Nutzeroberfläche nicht nur schnell von der Hand gehen, sondern auch komfortabel sein. Und wer die Auswahl aus einem grafisch dargestellten Dateibaum (z. B. in Pctools) kennt, braucht auch hier darauf nicht zu verzichten. Die Tastenkombination <Alt>-<F10> liefert einen solchen – der Nutzer kann mit den Cursortasten auswählen. Es sollte also ab jetzt kein Problem mehr sein, in beiden Panels jeweils das Laufwerk und den Pfad Ihrer Wahl einzustellen.
Zum Starten von Programmen brauchen Sie nur – sofern Sie nicht den Aufruf in der Kommandozeile bevorzugen – den Balkencursor auf dem Dateinamen zu positionieren und <Enter> zu drücken. Für die Zeit der Ausführung ist der Norton Commander nicht verfügbar, anschließend meldet er sich – wie oben beschrieben – wieder. Sollen in der Kommandozeile noch weitere Angaben gemacht werden (z. B. Parameter übergen), so kopiert man den Dateinamen einfach mit <Ctrl>-<Enter> in die Kommandozeile und ergänzt dann.

Funktionen

Der Norton Commander verfügt über einen integrierten Editor. Man braucht nur auf einer Textdatei zu positionieren und ist mit der Funktionstaste <F4> im Editiermodus. Die einfachen Funktionen sind Wordstar-kompatibel, leider existieren die Blockoperationen aber nicht. Einzig eine Suchoperation ist vorhanden (Funktionstaste <F7>). Die nicht in der Hilfszeile aufgeführte Suchwiederholung mit dem gleichem Begriff erreichen Sie mit <Shift>-<F7>. In der Kopfzeile werden die Zeilen, Spalten und freier Speicherplatz angezeigt. Dateien bis zu einer Größe von 30512 Byte können unter MS-DOS 3.30 und einem Hauptspeicher von 640 KByte bearbeitet werden. Ein kleines Extra finden Sie an der äußersten rechten Position, wenn diese nicht durch ein residentes Clock-Programm überdeckt ist: den ASCII-Wert des Zeichens, auf dem der Kursor gerade steht bzw. die Abkürzungen für End of Line und End of File. Das Sternchen zeigt an, ob am Text Änderungen vorgenommen wurden und ob deshalb beim Verlassen ein eventuelles Sichern hinterfragt werden muß. Eine Unterteilung in Seiten erfolgt nicht – Ziel ist die Programmerfassung und nicht die Textverarbeitung.
Die meisten Funktionen erschließen sich dem Nutzer leicht oder sind eh bekannt. Deshalb sei es erlaubt, auch auf ein paar Optionen einzugehen, die nicht so geläufig sind. Zum Beispiel darauf, daß es möglich ist, im integrierten Editor auch nicht druckbare Zeichen zu verwenden. Dazu ist <Ctrl>-<Q> einzugeben, und anschließend kann wie auf Kommandoebene mit <Alt> und dem Kode auf dem Ziffernblock das entsprechende Steuerzeichen erzeugt werden. In diesem Zusammenhang sei an die Schwierigkeiten bei der Verwendung von ANSI-Sequenzen in den letzten Kurs-Folgen erinnert. Und noch etwas: Viele Nutzer greifen immer noch auf **copy con datei** zurück, wenn sie eine neue Datei anlegen wollen. Versuchen sie es statt dessen einmal mit <Shift> <F4>. Norton fragt dann nach einem Dateinamen und erzeugt – falls nicht vorhanden – eine neue Datei. Ist Ihnen der Build-in-Editor zu spartanisch ausgestattet, so kann statt dessen ein beliebiger Ersatz (z. B. der Norton Editor oder Wordstar) – wie später bei **Setup** gezeigt – integriert werden.
Neben der Edit-Funktion (siehe auch Hilfe auf der untersten Bildschirmzeile) gibt es noch **View** zum Ansehen von Dateien (Funktionstaste <F3>). Es erlaubt die Anzeige weit größerer Dateien, als der Editor aufnehmen könnte und bewahrt Sie vor versehentlichen Änderungen. Verlassen wird dieser Modus – wie auch der Editor – mit <ESC>.
Neben der obligaten Hilfe auf <F1> zeigt die KeyBar genannte untere Bildschirmzeile noch Funktionen zum Kopieren, Umbenennen und Löschen. Sie beziehen sich im einfachsten Fall auf die Datei, auf der der Balkencursor gerade steht. Drücken Sie also <F5>, so wird nach dem Ziel des Kopiervorganges gefragt. Vorgabe ist stets das im anderen Fenster eingestellte Verzeichnis, sie kann aber beliebig abgeändert werden. Beachten Sie dabei bitte folgendes: Wird als erstes eine Bewegung mit den Kursortasten ausgeführt, so bleibt die Vorgabe stehen und kann verändert werden; schreiben Sie zuerst ein Zeichen, so verschwindet die Vorgabe ganz, und es wird eine komplette neue Eingabe erwartet. Die Angabe des Kopierzieles unterliegt den gleichen Regeln wie beim COPY-Befehl.
Mit <F6> erhalten Dateien einen neuen Namen. Weiterhin ist ein Verschieben möglich, sofern ein anderes Zielverzeichnis festgelegt wird. Verschieben bedeutet in diesem Falle Kopieren und anschließendes Löschen der Quelldatei, auch beim Verschieben von Diskette auf Festplatte und umgekehrt, was mitunter sehr hilfreich aber auch verhängnisvoll sein kann. Bei Löschen mit <F8> ist zu beachten, daß auch schreibgeschützte Dateien nach Rückfrage und Bestätigung mit der Enter-Taste gelöscht werden. Ein wenig Aufmerksamkeit ist also vonnöten.
Wie der Hilfszeile zu entnehmen ist, werden mit <F7> neue Verzeichnisse angelegt, aber das ist mit **md** wohl genauso schnell zu bewerkstelligen.
Der Norton Commander bietet auch die Möglichkeit, Dateien in Verzeichnissen zu behandeln, in denen man sich gerade nicht befindet. Dazu sind die eben beschriebenen Funktionen zusammen mit <Shift> zu verwenden. Wenn Sie <Shift>-<F5> drücken, so werden Sie nach Quell- *und* Zieldatei gefragt. Desgleichen auch für das Umbenennen und Löschen und – wie oben bereits erwähnt – für das Editieren von Dateien.
Noch gar nicht erklärt wurde, wie man mehrere Dateien zugleich behandelt. Dazu müssen diese markiert werden. Das geschieht mit <Ins> bzw. <Einfg> für jeweils die Datei, auf der der Balkencursor steht. Auf einem Farbdisplay ist die Markierung gut zu erkennen, im Schwarz/Weiss-Modus wird sie durch erhöhte Helligkeit oder durch einen Balken neben dem Dateinamen angezeigt. Mit erneutem Drücken von <Ins> kann – wenn der Balken auf der markierten Datei steht – die Markierung wieder aufgehoben werden. Eine weitere Erleichterung stellt die Markierung mit einer Dateimaske und die Verwendung der Joker dar. Dazu drücken Sie die Plus-Taste neben dem rechten Ziffernblock (zum Aufheben der Markierung die darüber liegende Minus-Taste) und es erscheint eine Maskenvorgabe, in diesem Fall *.* für alle Dateien. Es steht Ihnen frei, statt dessen eine beliebige Dateigruppe einzutragen. Anschließend kann für alle diese Datei eine Aktion ausgeführt werden.
Besonders hilfreich ist die Arbeit mit markierten Dateien, wenn auf mehrere Disketten kopiert werden soll. Ist eine Diskette voll, so ist ein Wechseln möglich, denn nur die verbleibenden Dateien sind noch markiert. Wollen Sie dagegen einmal ein ganzes Verzeichnis löschen, das noch dazu sehr viele Dateien enthält, so sind Sie gut beraten, dies mit del *.* zu tun, da eine solche Aktion im Norton Commander – bedingt durch das einzelne Anzeigen jeder Aktion – recht viel Zeit in Anspruch nimmt.
Immer wieder sucht man auf der Festplatte nach Dateien, von denen man nicht genau weiß, in welchem Verzeichnis sie stehen. Mit <Alt>-<F7> kann veranlaßt werden, nach einer Datei oder einer Gruppe von Dateien (bei Angabe einer Maske) auf dem aktuellen Laufwerk zu suchen. Alle gefundenen Dateien werden in einem Fenster angezeigt und können mit den Cursortasten ausgewählt werden.
Wer oft auf der Kommandoebene arbeitet, wird es zu schätzen wissen, in der Kommandozeile zurückblättern zu können – eine Option, die Ihnen vielleicht aus dBase oder vom Public-Domain-Programm Dosedit geläufig ist. Leider sind die Pfeiltasten bereits zur Bewegung des Balkenkursors im Verzeichnisfenster vergeben. Sie können aber mit der Ctrl-bzw. Strg-Taste die Tastenkodes erzeugen, die denen der Pfeiltasten entsprechen, nämlich <Ctrl>-<E> für aufwärts und <Ctrl>-<X> für abwärts. Mit diesen beiden Tastenkombinationen ist es möglich, im

Kommandostapel zu blättern, das heißt die zuletzt eingegebenen Befehle wieder in die Kommandozeile zurückzuholen und gegebenenfalls zu korrigieren. Zusätzlich bewahrt der Norton Commander die letzten Kommandoeingaben in einem separaten Fenster (History genannt) auf. Sie können es mit <Alt> <F8> aktivieren und anschließend menügeführt auswählen. Leider sind keine Veränderungen am Befehl mehr möglich.
Häufig verdeckt der Norton Commander wichtige Teile des Bildschirms. Abhilfe schafft die Möglichkeit, den gesamten Commander oder einzelne Teile auszublenden. MIt <Ctrl>-<F1> läßt sich das linke mit <Ctrl>-<F2> das rechte Panel aus- und wieder einblenden. Gleiches gilt für die Hilfszeile am unteren Bildschirmrand mit <Ctrl>-<B>. Für die meisten Aktionen ist ein Verzeichnisfenster ausreichend, das man zweckmäßigerweise auf der rechten Seite stehen läßt und so freien Blick auf die Ausgaben der gestarteten Programme hat.
Die Tastenkombination <Ctrl>-<O> blendet beide Panels aus. Beachten Sie aber bitte, daß der Norton Commander nicht nur im Speicher, sondern auch weiterhin aktiv bleibt, wie Sie am Betätigen der Funktionstasten leicht erkennen können. Zusätzlich kann jetzt auch mit den Pfeiltasten in der Kommandozeile geblättert werden.
Zu beachten ist, daß der Norton Commander beim Laden nicht prüft, ob er sich bereits im Speicher befindet. Er läßt sich also leicht ein zweites Mal laden, wenn Sie ihn nach <Ctrl> <O> erst einmal aus den Augen verloren haben. Spätestens beim vierten Mal ist dann aber der 640-KByte-Speicher voll.
Nicht vergessen wollen wir zu guter Letzt, wie man den Commander wieder aus dem Speicher entfernt. Wie anfangs erwähnt, wird es in erster Linie Speicherplatzmangel sein, der Sie dazu bewegt. Beim Verlassen mit <F10> wird noch einmal rückgefragt, um ein versehentliches Betätigen der Taste korrigieren zu können. Beachten Sie, daß alles, was Sie eingestellt haben, damit verlorengeht. Es sei denn, die Konfiguration wird mit <Shift>-<F9> in die Datei Nc.ini gespeichert. In diesem Falle findet der Nutzer beim erneuten Start alles so vor, wie es beim letzten Mal verlassen wurde. Wurde ein Fenster oder die Hilfszeile ausgeblendet, so wird auch das berücksichtigt.

Konfigurieren (Setup)

Der Norton Commander bietet weiten Freiraum zur Anpassung an die eigenen Wün-

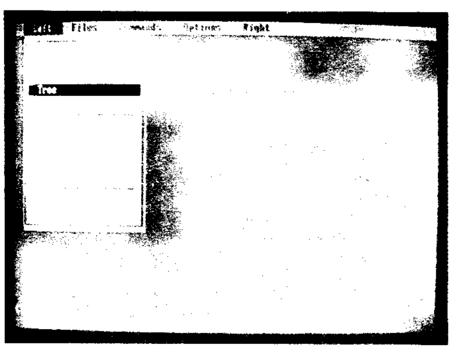


Bild 2 Einstellen des linken Panels im Setup-Menü

sche und Erfordernisse. Das betrifft in erster Linie das Outfit, also die äußere Erscheinung – in begrenztem Umfange aber auch die Funktionalität. Sämtliche Einstellungen werden über das Setup-Menü vorgenommen, das Sie mit <F9> erreichen. Die am oberen Bildschirmrand erscheinende Menüzeile enthält 5 Einträge, die für jeweils ein Pull-down-Fenster stehen. Die beiden äußeren Menüs sind für die Einstellung der Panele zuständig (siehe Bild 2). Mit den oberen fünf Einträgen wird festgelegt, was im Verzeichnisfenster angezeigt werden soll. **Brief** stellt die Anzeige der Dateien in Kurzform (ohne Größe und Zeit) ein, wie in Bild 3 rechts zu sehen. Hinter **Full** verbirgt sich die vollständige Anzeige, die wir aus Bild 1 kennen. Statt der Verzeichniseinträge kann auch der Verzeichnisbaum grafisch dargestellt werden, wie in Bild 3 links zu sehen. Dazu ist **Tree** zu wählen. Weiterhin können mit **Info** Informationen zum Hauptspeicher (gesamt, belegt) und zum aktuellen Laufwerk (Kapazität, frei) zur Anzeige gebracht werden. Da man diese nicht ständig benötigt, gibt es einen Shortcut zum zeitweiligen Umschalten, nämlich <Ctrl>-<L>. Warum ist diese Tastenkombination im Setup-Menü nicht aufgeführt?
Die mittleren fünf Punkte in Bild 2 betreffen die Reihenfolge, in der die Dateien angezeigt werden sollen: Sortiert nach Namen, Typ, Zeit und Größe. Beachten Sie aber, daß diese Sortierung nur die Anzeige betrifft und nicht etwa die Änderung der Einträge im Verzeichnis der Diskette, wie das beispielsweise Pctools tut. Die Tastenkombinationen <Ctrl>- <F1> zum Ausschalten des Panels sowie <Alt>-<F1> zum Wechseln des Laufwerks im linken Panel haben wir bereits kennengelernt, für das rechte Panel gilt entsprechend <F2>, was Sie auch im rechten Pull-down-Menü einstellen können.
Die Option **Re-read** veranlaßt, daß nach einem Diskettenwechsel die Einträge im Verzeichnisfenster aktualisiert werden. Das gleiche erreicht man aber schneller durch nochmaliges Anwählen des Laufwerks mit **a:** <enter> oder mit <Ctrl>-<R>.
Die Version 3.0 kennt darüber hinaus eine Filter-Funktion, die nur die durch eine Maske vorgegebenen Dateien anzeigt. Der Unterpunkt **Files** im Setup-Menü erlaubt noch einmal das interaktive Auswählen aller Funktionen, die auch in der Hilfszeile stehen und die wir bereits besprochen haben. Ein paar zusätzliche Einstellungen finden wir aber im **Option**-Menü (siehe Bild 4). In diesem Menü werden Schalter gestellt, die jeweils zwei Zustände annehmen können, was an einem kleinen Häkchen zu erkennen ist. Mit Color kann zwischen Farb- und Monochromdisplay unterschieden werden. Die Erfahrung zeigt, daß für eine Reihe von Schwarzweiß-Bildschirmen die Festlegung als Laptop die besten Kontraste bringt. Das betrifft besonders die Darstellung der markierten Dateien.
Der Punkt **Auto menus** veranlaßt, daß beim Start des Norton Commanders stets das nutzereigene Menü automatisch aktiviert wird, welches sonst unter <F2> erreichbar ist. Eine Maßnahme besonders für unbedarfte Nutzer.
Mit **Path prompt** kann die Anzeige des aktuellen Pfades im Prompt ein- und ausgeschal-

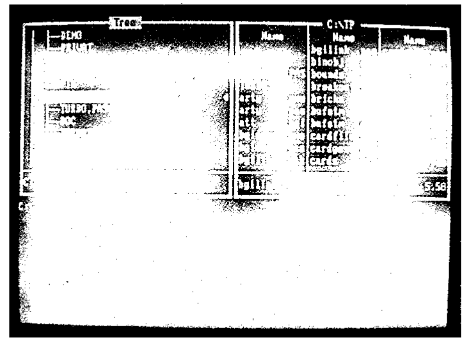


Bild 3 Anzeige im Brief-Format mit Dateibaum

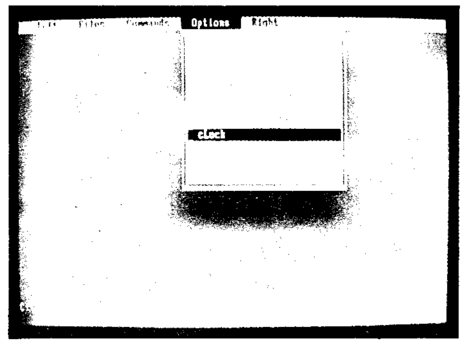


Bild 4 Das Option-Menü

tet werden. Man kann auf die Pfadanzeige durchaus verzichten, da an der Kopfzeile jedes Panels das aktuelle Verzeichnis ablesbar ist. Das Ausblenden der Hilfszeile (KeyBar) durch <Ctrl>-<B> ist Ihnen bereits vertraut. Die Verzeichnisfenster können auf den ganzen Bildschirm ausgedehnt (wie in Bild 1) oder nur auf halbe Höhe gezogen werden (wie in Bild 3). Dafür ist der Punkt **Full Screen** zuständig. **Mini status** bezieht sich auf die unterste Zeile der Panels in den Bildern 1 und 3. In dieser wird die aktuelle Datei vollständig angezeigt (wichtig, wenn mit der Kurzform gearbeitet wird). Wertvoll ist der Status besonders, wenn Dateien markiert werden – zeigt er doch die Gesamtgröße. Ebenfalls damit im Zusammenhang steht der Punkt **Ins move down**. Hier wird festgelegt, ob beim markieren mit <Ins> bzw. <Einfg> der Balkencursor automatisch eine Zeile nach unten soll. Wenn Sie die Uhrzeit gern stets im Blick haben, so kann diese mit **cLock** in der rechten oberen Ecke eingeschaltet werden. Das groß geschriebene L bedeutet, daß auch die Großbuchstaben in der jeweiligen Zeile zur Auswahl herangezogen werden können. Für alle, die statt des integrierten Editors lieber ein eigenes Textverarbeitungsprogramm auf <F4> legen wollen, gibt es den Menüpunkt **Editor**. Er erlaubt die Einbindung eines Textprogramms eigener Wahl, das dann mit <F4> erreichbar ist. Der eingebaute Editor bleibt aber weiterhin über <Alt>-<F4> erhalten. Wenden wir uns als nächstes dem Punkt **Commands** im Setup-Menü zu (siehe Bild 5). Die ersten fünf Eintragungen sind bis auf <Alt>-<F9> zum Umschalten auf den 43-Zeilen-Modus einschließlich ihrer Shortcuts bereits bekannt. Mit **Compare directories** können die Einträge zweier Verzeichnisse verglichen wer-

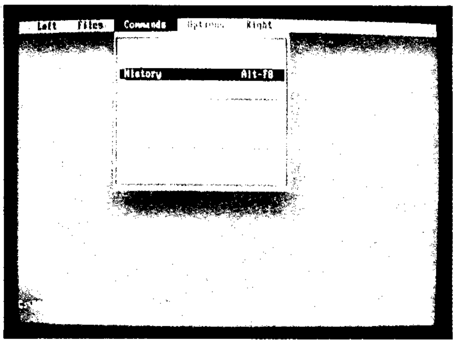


***Bild 5** Im Kommando-Menü sind die Shortcuts noch einmal zusammengefaßt*

den, etwa um festzustellen, ob beim Kopieren nichts vergessen wurde. Richtig interessant wird es aber bei den letzten beiden Menüpunkten. Sie erlauben das Anlegen und Editieren von Textfiles mit nutzereigenen Aktionen.
Für sich stets wiederholende Eingaben wird dem Nutzer die Möglichkeit gegeben, eigene Menüs zu definieren. Wenn Sie einmal <F2> bedienen, werden Sie feststellen, ob bereits derartige Vereinbarungen in einer Datei Nc.mnu getroffen worden sind und die Auswahl von vordefinierten Aktionen gestatten. Wenn nicht, können Sie sich diese selbst anlegen, wofür der Menüpunkt **Menu file edit** steht. Der Aufbau sieht folgendermaßen aus: Zuerst steht die jeweilige Menüausschrift (muß in der ersten Spalte beginnen), und auf den nächsten Zeilen (etwas eingerückt) folgen die dazugehörigen Kommandos. Hier als Beispiel einmal zwei Menüpunkte:

```
Testprogramm übersetzen
  cd\masm
  masm test;
  link test;
  exe2bin test test.com
  del test.obj
```

Das Beispiel zeigt, daß es sich auch für eine einmalige Testphase lohnen kann, einen Menüpunkt einzurichten. Beim Aufruf von Programmen sollte von der Möglichkeit Gebrauch gemacht werden, einen neuen Pfad einzustellen und Environment-Variablen zu setzen. Vor der Menüausschrift kann auch noch ein Buchstabe oder die Bezeichnung einer Funktionstaste (F1 ... F10) festgelegt werden und dann zur Kurzanwahl dienen. Tragen sie beispielsweise
T: Testprogramm übersetzen
ein, so ist dieser Punkt im Usermenü auch kurz mit **t** aktivierbar.
Ähnlich einfache Möglichkeiten der Ausführung von Nutzerkommandos bietet das sogenannte Extension-File (Datei Nc.ext). Hier wird festgelegt, was mit Dateien eines bestimmten Typs getan werden soll, wenn der Nutzer den Balkenkursor auf sie positioniert und <Enter> drückt. Für Exe-, Com- und Bat-Files sind bereits Aktionen festgelegt: Sie werden gestartet. Alle anderen Typen stehen für eigene Festlegungen zur Verfügung.
Die Eintragungen besitzen folgende Syntax:
Erweiterung : Kommando
Als Platzhalter für die aktuell ausgewählte Datei steht dabei ein Ausrufezeichen. Es wird ähnlich benutzt wie die bekannten Joker. So wird für **!** die aktuelle Datei ohne Erweiterung eingesetzt, für **!.!** die Datei mit Erweiterung. Für das aktuelle Laufwerk steht **!:** und für den aktuellen Pfad **!**.
Um also Pascal-Dateien mit dem Typ Pas durch den Kommandozeilencompiler übersetzen zu lassen, ist
pas: tpc !.!
einzutragen. Soll statt dessen die integrierte Entwicklungsumgebung aufgerufen werden und die betreffende Datei geladen werden, so erfordert das beispielsweise den Eintrag:
pas: c:\tp\turbo !:!\!.!
Sowohl die Datei Nc.mnu als auch Nc.ext kann auch mit jedem anderen Texteditor erzeugt werden. Zu erwähnen bleibt noch, daß der Norton Commander nach einer Menüdatei zuerst stets im aktuellen Verzeichnis sucht, so daß jedem Nutzer seine ganz persönliche Einstellung zur Verfügung steht.

Norton und die Maus

Der Norton Commander kann – wie man das von einem modernen Programm erwartet – auch mit der Maus bedient werden. Ob man die Mausbedienung bevorzugt, ist Gewohnheits- und Geschmackssache. Voraussetzung für die Nutzung des kleinen Nagers ist das Laden des Microsoft-Maustreibers. Dazu ist die Datei Mouse.com zu starten. Der Mauskursor ist anschließend als Rechteck (auf einem Farbbildschirm andersfarbig) zu erkennen. Positioniert man ihn auf einem Eintrag des Verzeichnisfensters, so kann dieser durch Drücken der linken Maustaste ausgewählt werden. Das entspricht der Bewegung des Balkenkursors mit den Pfeiltasten. (Anmerkung: Im folgenden wird stets auf die originale Zweitastenmaus von Microsoft Bezug genommen, daneben gibt es auch Exoten mit einer oder drei Tasten.) Die rechte Maustaste wirkt wie <Ins> bzw. <Einfg> und <Del> bzw. <Entf> die Dateien, auf denen der Kursor gerade steht, werden markiert, bzw. die Markierung wird aufgehoben. Sichtbar wird das wie beim Markieren über die Tastatur. Was die Enter-Taste auf der Tastatur, ist der Doppelklick (zweimaliges Drükken kurz hintereinander) der linken Taste für die Maus. Er dient zum Starten von Programmen (bzw. zum Ausführen einer Aktion entsprechend der Programmerweiterung). Mit der Maus kann auch sehr schnell gescrollt werden. Dazu ist der Mauskursor auf die oberste Zeile (die die Überschrift enthält) oder auf den unteren Rand zu setzen und die linke Taste zu drücken. Die Dateieinträge rollen automatisch nach oben bzw. unten. Wird statt dessen die rechte Taste festgehalten, so erfolgt gleichzeitig ein Markieren aller Dateien. Das geht sogar schneller als mit der Tastatur. Fährt man mit dem Kursor in die oberste Zeile, so erscheint automatisch das Setup-Menü (wie bei <F9>), und man kann per linkem Klick die einzelnen Optionen einstellen. Des weiteren ist ein Aktivieren der Menüpunkte am unteren Bildschirmrand (KeyBar genannt) durch einfaches Anklicken möglich.
Die Version 3.0 des Norton Commanders wartet mit einer Reihe weiterer Verbesserungen auf. Zu nennen sind die neuen Komponenten **Commander Link** (zur Verbindung von Rechnern über ein spezielles Kabel, z. B. zum „Füttern" eines Laptops), **Commander Mail** (zum Verschicken von Nachrichten im Netz) sowie **Quick View** zur direkten Unterstützung einer Vielzahl von internen Dateiformaten der gängigen Datenbank-, Kalkulations-, Textverarbeitungs- und Grafikprogramme.
Der Punkt **Configuration** im Option-Menü faßt grundsätzliche Einstellungen zusammen. Positiv anzumerken ist die Hilfe zu jedem Punkt mit <F1>, auffällig auch das schonende Verwandeln des Bildschirms nach einer bestimmten Zeit in einen dunklen Nachthimmel mit ständig aufgehenden und wieder verlöschenden Sternen.

Tafel 1 Tastenkombinationen des Norton Commanders

Taste	Funktion
<ESC>	Löschen der Kommandozeile, Abbruch einer Aktion *)
<Tab>	Wechseln des Verzeichnisfensters
<+>	Markieren von Dateigruppen
<–>	Aufheben von Gruppenmarkierungen
<PgUp>/<Bild↑>	Eine Seite nach oben blättern
<PgDn>/<Bild↓>	Eine Seite nach unten blättern
<Home>/<Pos1>	Balkenkursor auf ersten Eintrag
<End>,<Ende>	Balkenkursor auf letzten Eintrag
<Ins>/<Einfg>	Markieren einer Datei bzw. Aufheben der Markierung
<F1>	Hilfe
<F2>	Aufruf UserMenü
<F3>	Ansehen (View) von Dateien
<F4>	Editieren von Dateien
<F5>	Kopieren von Dateien
<F6>	Umbenennen oder Verschieben
<F7>	Anlegen von Verzeichnissen
<F8>	Löschen von Dateien oder Verzeichnissen
<F9>	Einstellen der Konfiguration (Setup)
<F10>	Verlassen des Norton Commanders
<Shift>-<F3>	Ansehen von Dateien mit Abfrage
<Shift>-<F4>	Anlegen einer neuen Datei
<Shift>-<F5>	Kopieren von Dateien mit Abfrage
<Shift>-<F6>	Umbenennen mit Abfrage
<Shift>-<F8>	Löschen von Dateien mit Abfrage
<Shift>-<F9>	Speichern der Konfiguration
<Alt>-<F1>	Laufwerkswechsel im linken Fenster
<Alt>-<F2>	Laufwerkswechsel im rechten Fenster
<Alt>-<F3>	Ansehen von dBase-Dateien ohne Dbview.exe
<Alt>-<F4>	Editieren mit Build-in-Editor
<Alt>-<F7>	Dateien suchen
<Alt>-<F8>	Auswahl von Befehlen aus dem History-Stapel
<Alt>-<F9>	Umschalten auf EGA-Modus (43 Zeilen)
<Alt>-<F10>	Anzeige des Dateibaums
<Ctrl>-<Break> /<Strg> <Untbr>	Abbrechen von Gruppenfunktionen
<Ctrl>-<F1>	Aus-/Einblenden des linken Fensters
<Ctrl>-<F2>	Aus-/Einblenden des rechten Fensters
<Ctrl>-<Enter>	Kopieren des Dateinamens in die Eingabezeile
<Ctrl>-<PgUp>	Wechsel ins übergeordnete Verzeichnis
<Ctrl>-<\>	Wechsel ins Hauptverzeichnis
<Ctrl>-<A>	Kursor ein Wort nach links *)
<Ctrl>-<B>	Hilfstexte (KeyBar) aus- bzw. einblenden
<Ctrl>-<E>	Rückwärtsblättern im Eingabepuffer *)
<Ctrl>-<F>	Kursor ein Wort nach rechts *)
<Ctrl>-<G>	Zeichen an Kursorposition löschen *)
<Ctrl>-<H>	Zeichen löschen links *)
<Ctrl>-<I>	wie <Tab>
<Ctrl>-<J>	wie <Ctrl>-<Enter>
<Ctrl>-<L>	Statusfenster ein- bzw. ausschalten
<Ctrl>-<O>	Aus- bzw. Einblenden beider Verzeichnisfenster
<Ctrl>-<P>	Aus- bzw. Einblenden des nicht aktiven Verzeichnisfensters
<Ctrl>-<T>	Wort rechts löschen *)
<Ctrl>-<U>	Vertauschen der Verzeichnisfenster
<Ctrl>-<W>	Wort löschen links *)
<Ctrl>-<X>	Vorwärtsblättern im Eingabepuffer *)
<Ctrl>-<Y>	Löschen der Kommandozeile *)

*)Aktionen beziehen sich auf die Kommandozeile

Connections

Teil 2

Alle wichtigen Rechner-Anschlüsse auf einen Blick

Carsten Meyer, Bernd Behr

Nach den mehr oder weniger normgerechten Floppy-, Drucker- und Kommunikationsschnittstellen in der letzten Kartei sind nun die mehr modellspezifischen Anschlüsse dran, angefangen beim DMA- und MIDI-Port des Atari bis hin zum Videostecker des Macintosh. Was sich sonst hinten in unzähligen Handbüchern versteckt, finden Sie hübsch übersichtlich auf den folgenden Seiten.

Atari

Der gemeine ST ist bei Musikern nicht umsonst wegen seiner schon serienmäßig eingebauten MIDI-Schnittstelle so beliebt. MIDI (Musical Instruments Digital Interface) wurden vor mehr als sieben Jahren von den Großen der Musikszene (u. a. Yamaha, Roland, Sequential Circuits, Kawai) genormt [1, 2] und beschreibt eine mit 31,25 kBd betriebene 5-mA-Stromschleife, die – über Optokoppler am Eingang potentialgetrennt – Daten der vielfältigsten Art an und von elektronischen Musikinstrumenten übermittelt. Abweichend vom Standard sind überzählige Pins der MIDI-Out-Buchse beim Atari als MIDI-Thru-Ausgang beschaltet, so daß man eventuell Ärger bekommt, wenn man hieran ein historisches „Diodenkabel" als MIDI-Verbindung mißbraucht.
Ein Störenfried in ganz anderer Größenordnung ist der sogenannte DMA-Port, auch ACSI genannt, der sich aufgrund seines verqueren Protokolls, der minimalen Treiberleistung und der hohen Empfindlichkeit gegenüber Störimpulsen unter laserdruckenden und festplattenspeichernden Atari-Freunden Feinde schafft. Trotz seiner nicht zu verleugnenden Ähnlichkeit zum SCSI-Bus bedarf es einigen Aufwands, hier SCSI-kompatible Peripheriegeräte anzuschließen [3].
Die etwas exotische Videobuchse des Atari führt eine Umschaltleitung, über die der Rechner erfährt, welcher Art der angeschlossene Monitor ist. Bei Farbdarstellung liefert sie ein RGB-Signal mit fernsehgerechten Synchronlängen, bei Monochromdarstellung dagegen 32 kHz Zeilen- und 70 Hz Bildfrequenz.

Macintosh

Der etwas andere Computer hat als einzige schnittstelle..echnische Besonderheit den noch an Apple-II-Zeiten erinnernden Floppy-Anschluß. Um beim Mac die gleichen Laufwerke wie bei der alten 3,5"-Doppel-Floppy-Station verwenden zu können, behielt man die dortige Pinbelegung bei. Andere als die speziellen Sony-Laufwerke kann man ohnehin nicht an den Mac anschließen, da nur diese die eingelegten Disketten selbst auswerfen können.
Einen Videoanschluß besitzen die kompakten Guckloch-Macs nur intern, bei den Modularen wird das analoge RGB-Signal über eine 15polige Submin-D-Buchse herausgeführt. Einige Macs (IIci) können über die Verschaltung der Masse-Pins (Kodierung) feststellen, welche Auflösung der angeschlossene Monitor besitzt.

Amiga

Die großzügig ausgelegte Videobuchse des Amiga 500/2000 dient nicht nur einem Monitor, sondern auch einem sogenannten Genlock-Interface als Anschluß, mit dem sich Videobilder und Computergrafiken synchronisiert mischen und überlagern lassen. Ferner führt die Buchse drei Versorgungsspannungen, die mit bis zu 300 mA belastet werden dürfen.

Kompatible

Die Tatsache, daß der PC abhängig von der eingebauten Steckkarte verschiedene Videonormen über denselben Buchsentyp ver-

Video PC/AT

9pol. Submin-D, weiblich (4)

Pin	Dig.-Mono Hercules	Dig.-RGBI CGA	Dig.-RGB EGA	Analog-RGB, Archimedes
1	Masse	Masse	Masse	Rot
2	Masse	Masse	Rot LSB	Grün
3	frei	Rot	Rot MSB	Blau
4	frei	Grün	Grün MSB	C-Sync/H-Sync
5	frei	Blau	Blau MSB	(V-Sync)
6	Intens.	Intens.	Grün LSB	Masse
7	Video	frei	Blau LSB	Masse
8	H-Sync	H-Sync	H-Sync	Masse
9	V-Sync	V-Sync	V-Sync	Masse

Video AT/VGA

15pol. HD-Submin-D, weiblich (15)
Analog-RGB

Pin	Funktion
1	Rot
2	Grün
3	Blau
4	Monitor ID Bit 2
5	n. c.
6	Rot-Masse
7	Grün-Masse
8	Blau-Masse
9	kein Loch (Verw.-Schutz)
10	Synchr.-Masse
11	Monitor ID Bit 0
12	Monitor ID Bit 1
13	H-Sync
14	V-Sync
15	n. c.

Video Atari ST

13pol. DIN (16)
Composite (FBAS), Analog-RGB, Ton

Pin	Funktion
1	Audio
2	neg. Comp.Sync-Signal
3	Port A, Bit 6, Soundchip
4	Monochrom-Sensor
5	Audio-Eingang
6	Grün
7	Rot
8	12 V (max. 10 mA)
9	H-Sync
10	Blau
11	Monochrom-Video
12	V-Sync
13	Masse

SCART-Buchse

21pol. Cenelec (18)
FBAS/Analog-RGB, Ton

Pins	Funktion
1	Audio Ausgang B
2	Audio Eingang B
3	Audio Ausgang A
4	Masse
5	Masse
6	Audio Eingang A
7	Blau (Analog 0,7 V_{ss})
8	Schaltspannung
9	Masse
10	n. c.
11	Grün (Analog 0,7 V_{ss})
12	n. c.
13	Masse
14	Masse
15	Rot (Analog 0,7 V_{ss})
16	Austastsignal
17	Masse
18	Masse
19	Video Ausgang 1 V_{ss}
20	Video Eingang 1 V_{ss}
21	Masse

Video Amiga 1000

8pol. DIN, weiblich (19)
Composite (FBAS), Ton

Pins	Funktion
1	n. c.
2	Masse
3	Audio links
4	Composite Video Output
5	Masse
6	n. c.
7	+12 Volt
8	Audio rechts

Video Amiga 500/2000/3000

23pol. Submin-D, männlich (17)
Analog-RGB, Digital-RGBI

Pin	Funktion
1	External Clock
2	External Clock Enable
3	Analog Rot
4	Analog Grün
5	Analog Blau
6	Digital Illuminance
7	Digital Blau
8	Digital Grün
9	Digital Rot
10	Composite Sync
11	H-Sync
12	V-Sync
13	Masse
14	Zero Detect
15	Clock Out
16	Masse
17	Masse
18	Masse
19	Masse
20	Masse
21	–5 Volt (50 mA)
22	+12 Volt (175 mA)
23	+5 Volt (300 mA)

Video Macintosh-Grafikkarten

15pol. Submin-D, weiblich (24) (25)
Analog-RGB

Pins	Funktion
1	Masse
2	Rot analog
3	C-Sync
3	Masse
5	Grün analog
6	Masse
7	n. c.
8	n. c.
9	Blau analog
10	n. c.
11	Masse
12	n. c.
13	Masse
14	Masse
15	n. c.

Video ist nicht gleich Video: Beachten Sie, daß einige Signale auch invertiert sein können.

MIDI-Schnittstelle Atari

5pol. DIN, weiblich (2)

Pin	MIDI-Out	MIDI-In
1	THRU Transmit Data	–
2	Abschirmung	–
3	THRU Loop Return (+)	–
4	OUT Transmit Data	IN receive Data
5	OUT Loop Return	IN Loop Return (+)

Vorsicht beim Atari: die MIDI-Out-Buchse ist doppelt belegt.

mittelt, führt beim Anfänger bisweilen zu böser Verwirrung. Analog-RGB-, TTL-RGBI- (mit Intensity-Leitung) und Monochrom-Monitore benutzen eine 9polige Submin-D-Steckverbindung, moderne hochauflösende VGA-Exemplare auch eine verwechslungsfreie 15polige Buchse.

Archimedes

Die Acorn-RISC-Maschinen bieten als Besonderheit einen Netzwerkanschluß. Die 5polige DIN-Buchse ist leider zu nichts anderem kompatibel als zum Acorn-eigenen Econet, das eine Übertragungsrate von 112 KBit/s schaffen soll.
Außer einer Analog-RGB-Buchse hat der Archimedes noch einen Anschluß für hochauflösende Monochrom-Monitore, die über zwei BNC-Stecker (BNC = beinahe no contact, Labornorm) angeschlossen werden. An der RGB-Buchse lassen sich die Pins 4 und 5 durch Jumper umkonfigurieren, so daß Pin 4 nicht mehr Composite-Sync liefert, sondern H-Sync; auf Pin 5 liegt dann das zugehörige V-Sync.

SCART

Diese an neueren Fernseh- und Videogeräten zu findende Buchse ist mit allerlei fernsehtypischen Signalen belegt. Die Analog-RGB-Leitungen sind optional und nur bei wenigen Geräten der Unterhaltungselektronik belegt. Ablenk- und Hilfsträgerfrequenzen sind in Deutschland der Farbfernsehnorm PAL untergeordnet (15,625 kHz Horizontal, 50 Hz Vertikal, 4,433 MHz FHT).

Farbig

Leider herrscht unter den Herstellern kaum Einigkeit über die Farbgebung der Stromversorgungsleitungen. Bei PC-Laufwerkskabeln und Netzteilen besteht meist folgende ‚Norm': rot – +5 V, schwarz – Masse, gelb – +12 V. Commodore dagegen belegt Gelb mit

Stromversorgung PC/AT

2 × 6pol. Messer intern (2)

Pin	P8	P9
1	Pwr Good (or)	Masse (sw)
2	–	Masse (sw)
3	+12 V (ge)	–5 V (ws)
4	–12 V (bl)	+5 V (rt)
5	Masse (sw)	+5 V (rt)
6	Masse (sw)	+5 V (rt)

Stromversorgung Atari

7pol. DIN (23)

Pin	Funktion
1	+5 V
2	n. c.
3	Masse
4	+12 V
5	–12 V
6	+5 V
7	Masse

Stromversorgung Laufwerke

Pin	Funktion	Farbe
1	12 V	(ge)
2	Masse	(sw)
3	Masse	(sw)
4	5 V	(rt)

Stromversorgung PS/2

15pol. AMP intern

Pin	Funktion
1	+5 V
2	Masse
3	+12 V
4	+5 V
5	Masse
6	Masse
7	+5 V
8	Masse
9	–12 V
10	+5 V
11	Masse
12	Pwr Good
13	+5 V
14	Masse
15	Sys Status

Stromversorgung Macintosh

	SE intern	Plus intern
	14pol. Amp	10pol. Pfosten
	(20)	(21)
Pin	**Funktion**	**Funktion**
1	+12 V	/VID
2	–12 V	/H-Sync
3	+5 V	Spkr
4	–5 V	/V-Sync
5	+5 V	+5 V
6	Masse	Masse
7	/V-Sync	–12 V
8	Masse	Masse
9	/H-Sync	+12 V
10	Masse	Batt.
11	/VID	
12	Masse	
13	Masse	
14	Masse	

Floppy Apple II GS, Macintosh

19pol. Submin-D, weiblich (22)

Pin	Funktion
1	Masse
2	Masse
3	Masse
4	Masse
5	– 12V
6	+5 V
7	+12 V
8	+12 V
9	n. c.
10	PWM Motor
11	PH0 Register Select 1
12	PH1 Register Select 2
13	PH2 Register Select 3
14	PH3 Register Write Strobe
15	/WREQ Write Data Request
16	SEL Register Select Line
17	/EN.EXT External Drive Enable
18	RD Read Data
19	WR Write Data

Archimedes Econet

5pol. DIN, weiblich (2)

Pin	Funktion
1	Data +
2	Masse
3	/Clock +
4	/Data –
5	Clock

Atari-ROM-Port

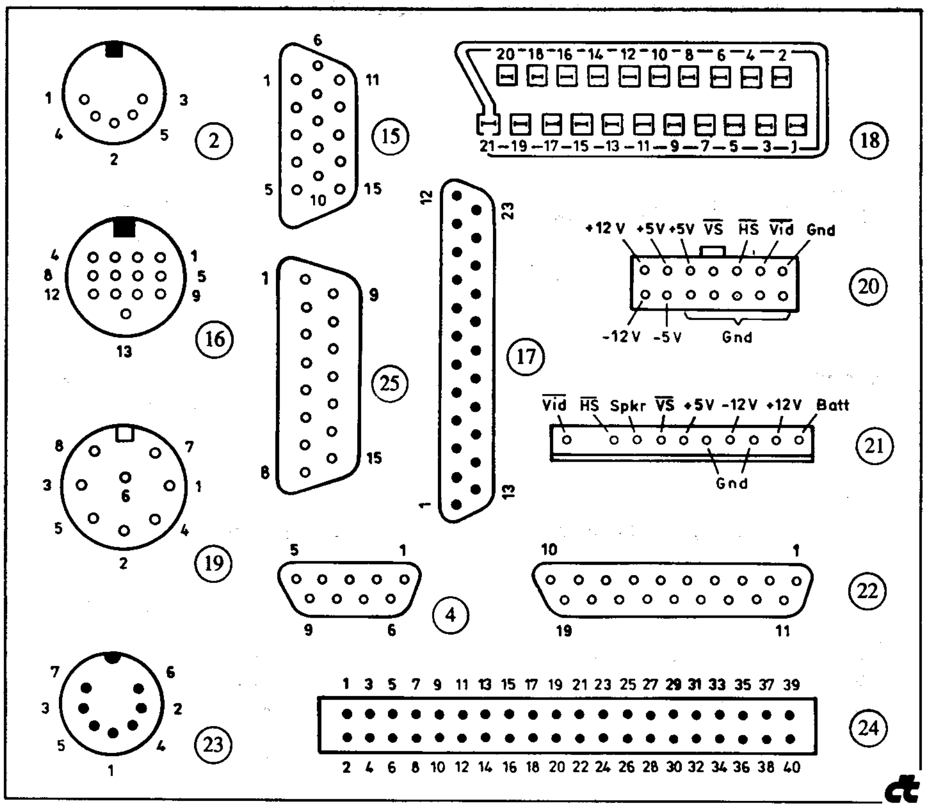
40pol. Pfosten (24)

Pin		Pin	
1	+5 V	21	A8
2	+5 V	22	A14
3	D14	23	A7
4	D15	24	A9
5	D12	25	A6
6	D13	26	A10
7	D10	27	A5
8	D11	28	A12
9	D8	29	A11
10	D9	30	A4
11	D6	31	ROMSel2
12	D7	32	A3
13	D4	33	ROMSel4
14	D5	34	A2
15	D2	35	UDS
16	D3	36	A1
17	D0	37	LDS
18	D1	38	Masse
19	A13	39	Masse
20	A15	40	Masse

Farbkennzeichnung

Abk. dtsch.	Abk. engl.	Farbe	Wertigkeit
sw	bk	schwarz	0
br	bn	braun	1
rt	rd	rot	2
or	or	orange	3
ge	ye	gelb	4
gn	gn	grün	5
bl	bl	blau	6
vi	pu	violett	7
gr	gr	grau	8
ws	wt	weiß	9
rs	pk	rosa	–

Pin-Nummern und Bilder beziehen sich auf die Steckverbinder am Gerät, auf die Kontakte gesehen.



+5 V und Orange mit +12 V. Blaue Kabel deuten auf eine negative Spannung hin (etwa −12 V im PC), während es sich bei allen anderen Farben ebensogut um eine Steuer- oder Meldeleitung (z. B. Power Good) handeln kann. (cm)

Nachdruck aus c't – Magazin für Computertechnik, Heft 9/1990

TERMINE

24. Fachkolloquium Informationstechnik

WER? Technische Universität Dresden, Sektion Informationstechnik
WANN? 19. bis 21. Februar 1991
WO? Technische Universität Dresden
WAS? Elektro- und andere Akustik, Elektronische und elektromechanische Meßtechnik, Sprach- und Objekterkennung
WIE Teilnahmemeldungen bitte an: Technische Universität Dresden, Sektion Informationstechnik, Bereich Akustik und Meßtechnik, Tagungsbüro, Mommsenstraße 13, O-8027 Dresden; Tel. 4634463 oder 4632244, Fax. 3751470294

Wö.

Längenbestimmung von Turbo-Pascal-Overlays

Rolf Schmieder, Berlin

Turbo-Pascal bietet (außer in der Version 4.0) die Möglichkeit der Verwendung von Overlays. Damit wird die Erarbeitung von Programmen möglich, die größer als der verfügbare Speicherraum sind. Diese Möglichkeit ist für viele Fälle völlig ausreichend. Probleme entstehen, wenn trotz mehrerer Overlay-Gruppen der verfügbare Speicherraum immer weiter abnimmt. Man erinnert sich dann an den Hinweis in der Literatur, man möge möglichst Overlays gleicher Größe in einer Gruppe unterbringen. Damit entsteht die Frage danach, welche Länge die einzelnen Overlays haben.

Allein aus dem Quelltext ist diese Frage schwer zu beantworten, da oft einfache, kompakte Strukturen einen weit größeren Speicherbedarf besitzen, als man ihnen ansieht. Die Lösung liegt in der Längenbestimmung aus dem Maschinenkode der Overlay-Gruppen (Bild 1).

Die Einbettung des Overlays im Kodebereich erfolgt entsprechend der Stelle seiner Erklärung im Quellcode.

Eine Overlay-Gruppe benötigt 5 Byte für den Aufruf, 11 Byte für den Namen und n-mal 128 Byte für den Kode.

Der Aufruf besteht aus einem Call-Befehl, der das Turbosystem ruft und dem zwei Byte – die Blocknummer im Overlay-File – folgen. An Hand dieser Nummer wird bei weiteren Overlay-Aufrufen kontrolliert, ob das geforderte Overlay aktuell geladen ist, und so ein Mehrfachladen verhindert. Danach stehen der Name des Hauptprogramms, der gegebenenfalls mit Leerzeichen auf 8 Byte verlängert wurde, und die dreistellige Nummer der Overlay-Gruppe (beginnend mit 000) im ASCII-Kode. Nun folgt ein Speicherbereich, dessen n-mal 128 Bytes den Wert 0 enthalten, wobei n die Anzahl von 128-Byte-Kodeblöcken des größten Overlays der Gruppe symbolisiert. Er nimmt die aufgerufenen Overlays auf.

Zum Aufruf eines Overlays werden an entsprechender Stelle im Programm die Register HL und DE mit der Blocknummer im Overlay-File und der Blockanzahl des Overlays geladen. Durch den folgenden Call-Befehl auf den oben erläuterten Aufruf des Overlays werden das Laden und die Ausführung des Overlays veranlaßt.

Die einzelnen Overlays stehen als Unterprogramme auf der Diskette oder der Festplatte, sind also mit dem Befehl Return (8 Bit: C9H; 16 Bit: C3H) abgeschlossen und belegen jeweils vollständige Sektoren. Overlays einer Gruppe stehen dabei hintereinander in einem File, wobei jedes Overlay an einem Sektorenanfang beginnt.

Dabei sind zwei Fälle zu unterscheiden:

① Der Maschinenkode des einzelnen Overlays ist kürzer als das Produkt aus Sektorlänge und Anzahl der belegten Sektoren. In diesem Fall ist der Rest des letzten Sektors nach dem abschließenden Return mit 00 gefüllt.

② Der Maschinenkode des einzelnen Overlays füllt genau die belegten Sektoren. In diesem Fall steht als letztes Byte C9H (bzw. C3H) im letzten Sektor.

Das Programm

Zunächst braucht in den einzelnen Overlay-Gruppen jeweils nur nach dem Auftreten von 00 oder C9H am Sektorende gesucht zu werden. Wird 00 gefunden, so muß im Sektor rückwärts nach C9H gesucht werden, wobei bis dahin nichts anderes als 00 gefunden werden darf. Gelingt das, ist fast sicher ein Overlay-Ende erreicht. Dieses Verfahren versagt leider in seltenen Fällen, wenn die Kombination C9 00 ans Sektorende gelangt.

Wird C9H am Sektorende gefunden, so kann es sich auch um das Ende eines Unterprogramms im Overlay handeln. Diese Entscheidung kann nur vom Programmierer an Hand der Anzahl und Größe der einzelnen Overlays getroffen werden, was jedoch im allgemeinen leicht möglich ist.

Das Programm versieht die gefundenen Overlays mit einer laufenden Nummer, gibt die Anzahl der belegten Sektoren, die Länge des Maschinenkodes in Byte und die im letzten Sektor noch freien Bytes an. Sollte C9H am Sektorende stehen, wird das durch ein Fragezeichen vermerkt. Die Längenbestimmung kann in diesem Fall wiederholt werden, wobei C9H am Sektorende nicht mehr als Overlayende betrachtet wird. Bei der Ausgabe auf Drucker wird durch Bildschirmdialog das gewünschte Verhalten ermittelt.

Die Längen der einzelnen Overlays sind zwar im Laufzeitsystem bekannt, stehen jedoch nicht an zentraler Stelle, so daß eine von den genannten Unwägbarkeiten unabhängige Ermittlung der Overlaylängen weitaus aufwendiger als im gezeigten Programm wäre.

Ist man gezwungen, das vorgestellte Programm häufiger zu verwenden, so kann man es leicht so erweitern, daß die Längenbestimmung für alle Overlays in verschiedenen Ebenen eines Programms automatisch erfolgt.

Eine weitere zweckmäßige Ergänzung besteht darin, eine ASCII-Datei zu erzeugen, die die Namen der Overlays enthält, die bei der Längenbestimmung dann benutzt werden, so daß an Stelle der numerierten Overlays die tatsächlichen Namen verändert werden.

```
PROGRAM overlayLaengen;          {Rolf Schmieder 15.1.1990}
VAR blockPos,                    {position im block}
    blockAnzahl,                 {des akt. overlays}
    OvNummer    : integer;       {zaehlnummer des overlays}
    bytes       : real;          {overlaylaenge in Byte}
    ovr         : FILE;
    ovrFile     : STRING[14];
    block       : ARRAY[0..127] OF byte;
    c,klar      : char;
    out         : TEXT;
    gefunden,                    {overlay-ende gefunden}
    unklar,                      {overlay-ende unklar}
    ignorieren,                  {unklare overlay-enden ignorieren}
    fileExistiert,
    LstFlag        : boolean;    {druckerausgabe}
BEGIN
gotoxy(1,1);  clrscr;            {fuer P8000 : write(#27':')}
writeln('LAENGENBESTIMMUNG VON TURBO-PASCAL - OVERLAYS  V 1.0');
klar := 'N';
REPEAT                           {Laengenbestimmung}
   REPEAT                        {Eingabe File-Name}
      fileExistiert := false;
      IF upcase(klar) = 'N' THEN
         BEGIN
         writeln;
         write('Name des Overlay-Files (oder ET fuer Ende) : ');
         readln(ovrFile);
         ignorieren := false;
         END
      ELSE
         ignorieren := true;  {Laengenbestimmung wiederholen}
      IF ovrFile <> '' THEN
         BEGIN
         IF NOT ignorieren THEN
            BEGIN
            write('Ausgabe auf Drucker ? (j/N) : ');
            LstFlag := false;
            read(kbd,c);  c := upcase(c);
            IF c = 'J' THEN    {Ausgabegeraet zuweisen}
               BEGIN
               writeln(c);
               LstFlag := true;
               assign(out,'CON:');
               END
            ELSE
               BEGIN
               writeln('N');
               assign(out,'LST:');
               END;
            rewrite(out);
            END;
         assign(ovr,ovrFile);
         {$I-} reset(ovr); {$I+}
         IF ioresult <> 0 THEN
            BEGIN writeln;
            writeln('File "',ovrFile,'" existiert nicht !')
            END
         ELSE
            fileExistiert := true;
         END;
   UNTIL fileExistiert OR (ovrFile = '');
   IF ovrFile <> '' THEN
      BEGIN
      writeln(out);
      writeln(out,'Overlay-Laengen in ',ovrFile,' :');
      writeln(out);
      unklar := false;  OvNummer := 1;
      REPEAT                 {until eof}
         klar := ' ';
         gefunden := false;   blockAnzahl := 0;
         REPEAT               {overlay-Ende suchen}
            blockread(ovr,block,1);
            blockPos := 127;
            blockAnzahl := succ(blockAnzahl);
            IF (block[127] = $c9) THEN {moegliches overlay-Ende}
               BEGIN
               c := 'J';
               IF LstFlag THEN
                  BEGIN
                  write('Overlay-Ende unklar ! (Block : ',
                        blockAnzahl,') Drucken (j/N) ? : ');
                  read(kbd,c);  c := upcase(c);
                  IF c <> 'J' THEN
                     c := 'N';
                  writeln(c)
                  END;
               IF (upcase(c) = 'J') AND NOT ignorieren THEN
                  BEGIN
                  klar := '?';        {als moeglich kennzeichnen}
                  unklar := true;
                  gefunden := true;
                  END
               END
            ELSE
               IF block[127] = 0 THEN
                  BEGIN
                  REPEAT
                     blockPos := pred(blockPos);
                  UNTIL (block[blockPos] <> 0) OR (blockPos = 0);
                  IF block[blockPos] = $c9 THEN
                     gefunden := true; {sicheres overlay-Ende}
                  END;
         UNTIL gefunden OR eof(ovr);
         bytes := 128.0 * (blockAnzahl-1)+ blockPos+ 1;
         write(out,'OVERLAY ',OvNummer:3,' ',klar,' : ',
               round(bytes):5,' Byte    (',
               round(128.0 * blockAnzahl - bytes):3,
                frei)     [',blockAnzahl:3,' Blo');
         IF blockAnzahl = 1 THEN
            writeln(out,'ck]')
         ELSE
            writeln(out,'ecke]');
         OvNummer := succ(OvNummer);
      UNTIL eof(ovr);
      klar := 'N';
      IF unklar AND NOT (ignorieren OR LstFlag) THEN
         BEGIN
         writeln;
         write('Wiederholung ohne " ? " (J/n) : ');
         read(kbd,klar);   klar := upcase(klar);
         IF klar <> 'N' THEN
            klar := 'J';
         writeln(klar)
         END;
      END;
UNTIL ovrFile = '';
END.
```

Bild 1 Das Programm OVL.PAS bestimmt die Längen von Turbo-Pascal-Overlays.

Das Programm wurde mit der Turbo-Pascal-Version 3.00 unter CP/M und MS-DOS erprobt.
Eine Anwendung des Programms auf 16-Bit-PCs erfordert allerdings, im Quelltext den Maschinenkode für Return von C9H auf C3H zu ändern.

Modulare Tabellensteuerprogramme

Wolfgang Franke, Plauen

Speicherprogrammierbare Steuerungen werden in zunehmendem Maße ein wichtiges Element der Automatisierungstechnik. Mit dem rasch größer werdenden Hardwareangebot wächst auch die Vielfalt der Software. Während die Hardwarepreise in Zukunft eher fallen werden, dürfte die Software erhebliche Kosten verursachen. Dies ist besonders dann ungüngstig, wenn man gezwungen ist, den Steuerungstyp häufig zu wechseln.
Für speicherprogrammierbare Steuerungen haben sich auch deshalb die weit verbreiteten Anwenderprogrammdarstellungen Anweisungsliste, Funktionsplan und Kontaktplan erhalten. Für den Kontaktplan ist dies auf die Allgemeinverständlichkeit der Programmbeschreibung, die dem Prinzip des Relaisschalterplanes nahekommt, zurückzuführen. Die Anweisungsliste ist an traditionelle Programmiertechnik angelehnt, wobei allerdings der funktionelle, strukturelle Zusammenhang aus der Darstellung schwer zu entnehmen ist; der Funktionsplan, der auf den Logikschaltkreissymbolen aufbaut, benötigt für eine gute Darstellung doch erhebliche grafische Mittel.
Das modulare Tabellensteuerprogramm stellt demgegenüber eine neue Lösung zur Programmierung von speicherprogrammierbaren Steuerungen dar. Die Tabellenform als Entwurfsgrundlage ist allgemeinverständlich und kommt der Denkweise von Konstrukteuren entgegen. Für die Entwicklung von modularen Tabellensteuerprogrammen ist kein spezielles Entwicklungssystem nötig, sondern die recht umfangreiche Entwicklungssoftware läuft auf jedem Personalcomputer, Bürocomputer oder ähnlichen Rechner. Auf der Zielhardware, der speicherprogrammierbaren Steuerung selbst ist dann nur ein kleiner Interpreter (Laufzeitmodul) erforderlich, um das Steuerungsprogramm abarbeiten zu können.

Entwicklungssoftware

Die Entwicklungssoftware funktioniert nach dem Turboprinzip, ähnlich dem bekannten Turbo-Pascal. Sie umfaßt Editor, Listengenerator und Debugger. Mit dem Editor wird das Anwenderprogramm erfaßt und bei der Eingabe gleich syntaktisch geprüft. Der Listengenerator erzeugt dann die eigentliche Steuerliste, und über Anfragen werden die konkreten Adreßbezüge ergänzt, die insbesondere dann wichtig sind, wenn das Programm aus mehreren Einzeltabellen besteht. Im Ergebnis entsteht ein Programm, das mit Hilfe des Interpreters auf der Zielhardware abgearbeitet werden kann. Der Debugger gestattet den Test des Steuerprogramms auf dem Entwicklungsrechner. Die Syntax des modularen Tabellensteuerprogramms ist äußerst einfach und übersichtlich. Das modulare Tabellensteuerungsprogramm wird in Tabellenform notiert. Die Tabelle besteht aus drei Teilen: dem Bedingungsteil, dem Aktionsteil und

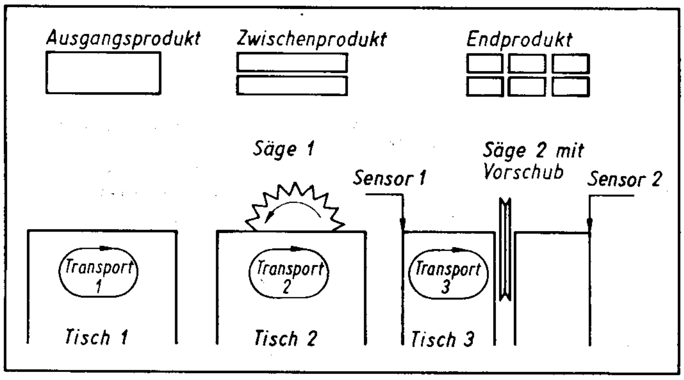


Bild 1 Beispiel Sägeanlage

Steuerschritt \ Bedingung/Aktion		Bedingung (Eingänge)					Aktion (Ausgänge)						
		Taste Start	Last Säge 1	Sensor 1	Sensor 2	Vorschubende Säge 2	Transport 1 Ein/Aus	Transport 2 Ein/Aus	Transport 3 Ein/Aus	Säge 1 Ein/Aus	Säge 2 Ein/Aus	Vorschub Säge 2 vor/zurück	Aktivierung
Start	1	1					1	1	1	1	1		2, 4
Vereinzeln	2		1				0						3
Nachschub Säge 1	3		0	0			1						2
Absägen	4				1				0			1	5
Nachschub Säge 2	5					1			1			0	4

dem Markierungsteil. Im Bedingungsteil werden die logischen Zustände der Eingabeleitungen aufgeschrieben, die vorliegen müssen, damit die Bedingung erfüllt ist. Bei den Eingabeleitungen kann es sich entweder um real existierende, physische Leitungen oder nur im Speicher nachgebildete, logische Leitungen handeln. Im Aktionsteil wird die Belegung der Ausgabeleitungen (physische oder logische), die hergestellt werden soll, wenn die dazugehörige Bedingung erfüllt ist, festgelegt. Im Markierungsteil werden im einfachsten Fall Zeilennummern aufgeschrieben, deren dazugehörigen Zeilen bei erfüllter Bedingung aktiviert oder deaktiviert werden. Daneben gibt es weitere Möglichkeiten, zum Beispiel eine Zeitüberwachung, die hier aber nicht näher ausgeführt werden soll. Die Tabelle enthält also in den einzelnen Zeilen einfache Wenn-dann-Formulierungen. Jede der Zeilen kann aktiv oder inaktiv sein. Nur aktive Zeilen werden tatsächlich abgearbeitet. Da sich Zeilen gegenseitig aktivieren und entaktivieren können, ist es möglich, komplexe Steuerungsabläufe zu programmieren.

Anhand eines sehr stark vereinfachten Beispiels soll die Programmentwicklung erläutert werden: Es sei eine Sägeanlage angenommen, die drei Prozeßschritte beinhaltet (Bild 1). Dabei besteht die Aufgabe darin, eine Folge dicht gepackter Bretter zu vereinzeln, längs zu teilen und danach mehrfach Stücke fester Länge abzuschneiden. Die beiden Sägen können nicht synchron arbeiten, da für die Arbeitsleistung der Säge 2 mehr Zeit benötigt wird. Sensor 1 gibt das Signal über das Ende des Sägeeingriffs am Zwischenprodukt. Sensor 2 bestimmt die Länge des Endproduktes. Das Vorschubende der Säge 2 wird ebenfalls signalisiert. An Säge 1 wird die Last am Motor festgestellt, und mit einem Taster kann die Anlage gestartet werden. Tafel 1 zeigt die Steuertabelle für die Lösung des Problems.

Im Grundzustand ist nur Zeile 1 aktiv. Solange die Taste nicht gedrückt ist (logisch 0), durchläuft der Interpreter die 5 dargestellten Zeilen immer wieder, findet nur die aktive Zeile 1 vor, deren Bedingung nicht erfüllt ist, und führt somit keine Aktion aus. Wird die Taste gedrückt, so werden die Ausgänge *Transport 1, 2 und 3* und *Säge 1 und 2* gleichzeitig zugeschaltet, und es werden die Zeilen 2 und 4 aktiviert. Die Zeile 1 verliert ihre Aktivierung (ob danach der Taster gedrückt oder nicht gedrückt ist, bleibt ohne Belang). Zeile 2 wartet darauf, daß der Sensor *Last Säge 1* aktiv wird. Gelangt ein Brett mittels Transport 1 zur Säge 1, so wird mit dem Beginn des Sägevorganges der Leistungsbedarf der Säge 1 ansteigen. Die Bedingung *Last Säge 1 aktiv* wird erfüllt und die Aktion *Abschalten von Transport 1* in Zeile 2 ausgeführt. Gleichzeitig wird Zeile 3 aktiviert. Der Sägevorgang wird auf diese Art und Weise zum Vereinzeln genutzt. Der Transport 2 bewirkt dann den weiteren Vorschub. Während das längsgeteilte Brett verlassen wurde, wird die UND-verknüpfte Bedingung in Zeile 3 erfüllt und der Transport 1 erneut zugeschaltet. Mit der gleichzeitigen Aktivierung der Zeile 2 beginnt wieder der Säge- und Vereinzelungsvorgang. Der in den Zeilen 4 und 5 beschriebene Quersägevorgang läuft davon völlig unabhängig in ähnlicher Weise ab und kann ohne weiteres nachvollzogen werden. Anhand dieses einfachen Beispiels sollte deutlich werden, wie die verschiedenen asynchronen und unabhängigen Prozesse in einer solchen Bearbeitungstabelle notiert werden. Die Tabelle wird in gleicher Form über den Editor auf dem Hostrechner eingegeben. Die Quelle wird durch einen Codegenerator in eine frei verschiebliche Liste übersetzt. Mit dieser Liste ist die Durchführung des Softwaretests auf dem Hostrechner möglich. Die gleiche Liste bildet die Grundlage für die Abarbeitung durch den Interpreter auf der Steuerungshardware.

Interpreter

Die Prozeßführung wird durch einen Interpreter gewährleistet. Dieser Interpreter wird nach der Initialisierung des Systems angesprungen und nicht mehr verlassen. Er führt die Bearbeitung der Steuerlisten durch, ruft gegebenenfalls besondere Programme (Sonderfunktionen) auf und übernimmt die Arbeit der Ein-/Ausgabe. Das Steuerprogrammsystem ist in seiner Einfachheit auf vorwiegend logische und zeitliche Aufgaben ausgerichtet. Das ermöglicht die Verwendung eines kompakten Interpreters (kleiner 1 KByte) und eröffnet damit die Nutzung des modularen Tabellensteuerprogramms sowohl für mittlere Steuerungen als auch für Einplatinenrechner mit einem begrenzten Speicher. Da auch die generierten Steuerlisten sehr kompakt sind (50 Zeilen etwa 1 KByte), ergeben sich trotz ihres geringen Speicherumfanges gute Leistungsparameter. Benötigt man dagegen eine höhere Leistungsfähigkeit bezüglich der Ein-/Ausgabeleitungen und/oder der Steuerungskapazität, so besteht die Möglichkeit, mehrere Tabellen bereitzustellen (folglich sind n x 128 E/A-Leitungen ansprechbar). Vom Interpreter wird diese Bearbeitungsform unterstützt, sie stellt damit eine weitere Ebene der Modularisierung dar. Die verschiedenen Tabellen werden je nach Belegung des *Tabellenaktivvektors* bearbeitet oder übergangen. Dies gewährleistet die Aktivierung und die Deaktivierung von Tabellen während des Programmlaufs entweder in einem hierarchischen System, wenn von einer Tabelle aus die Entscheidung getroffen wird (Verwaltung), oder in einem System gleichberechtigter Tabellen.

Der vorliegende Interpreter wurde für die Bearbeitung von 16 Tabellen ausgelegt. Die Zykluszeit (Zeit zwischen 2 E/A-Operationen) ist in dem modularen Tabellenprogramm variabel, das heißt, sie richtet sich nach den Aktivitäten innerhalb einer (oder mehrerer) Tabelle(n). Einen Durchschnittswert anzugeben ist sehr schwierig. Die kürzeste Zykluszeit bei einem Takt von 2,5 MHz liegt für 8 Zeilen, 8 Eingabeleitungen, 8 Ausgabeleitungen und eine aktive Zeile bei 1,5 ms; bei 50 Zeilen, 128 Ein-/Ausgabeleitungen und einigen aktiven Zeilen bei etwa 7 ms. Die Echtzeitfähigkeit für den Anwendungsfall ist an diesen Werten zu orientieren, wobei der Programmierer es besonders bei größeren Systemen in der Hand hat, die Aktivitäten in zeitkritischen Situationen kurzfristig zu reduzieren und somit auf bevorstehende Ereignisse schnell zu reagieren. Es muß ergänzt werden, daß modulare Tabellensteuerprogramme interruptfrei arbeiten, aber Interrupts, die der Anwender anderweitig benötigt, zulässig sind.

Schnittstellen

Ein Problem prinzipieller Art für höhere Programmiersprachen ist die Kommunikation mit der Umwelt. Dafür universelle Programme zur Verfügung zu stellen ist nahezu unmöglich. Ähnlich wie in einigen Hochsprachen wurde bei der Entwicklung des modularen Tabellensteuerprogramms diese Aufgabe erst einmal völlig ausgeklammert und dem Begriff *Treiber* zugeordnet. Als Schnittstellen wurden die „lokalen und globalen Vektoren" definiert. Dies sind Bytebelegungen im RAM, die ein vollständiges Abbild der Ein- und Ausgabeleitungen für die Einzelprozesse und den Gesamtprozeß geben. Der Treiber wird in die beiden Programmteile lokaler und globaler Treiber unterteilt. Dabei ist der globale Treiber allein für die Peripherie zuständig. Der lokale Treiber beinhaltet die Übergabe vom globalen Ein-/Ausgabevektor an den lokalen Ein-/Ausgabevektor, wobei diese an sich triviale Aufgabe bei Mehrtabellensystemen eine wesentliche Bedeutung erlangt. Werden für die Lösungen der Steueraufgabe Sonderfunktionen benötigt, so werden auch diese im lokalen Treiber plaziert. Übernimmt ein Anwender vom Steuerungs-

hersteller Hardware und Software, so braucht er sich üblicherweise um die hardwarenahe Programmierung nicht zu kümmern. Das modulare Tabellensteuerprogramm bietet mit der einfachen Tabellenerzeugung genau diese Möglichkeit. Dem Anwender steht es allerdings offen, auch in die Schnittstelle einzugreifen, so daß eigene Hardware oder Ergänzungen genutzt werden können. Die hohe Formalisierung des globalen und des lokalen Treibers bietet dazu die beste Voraussetzung.
Ein weiteres, wesentliches Kriterium für eine effiziente Programmentwicklung ist die einfache Änderbarkeit der Prozeßsteuerprogramme. Dabei ist es in der Entwicklungsphase notwendig, in sich geschlossene Programme, das heißt funktionssichere Programme, erzeugen zu können. Ein Programmabsturz sollte nicht möglich sein. Die syntaktische Richtigkeit, die bei der Editierung erzwungen wird, gewährleistet für das modulare Tabellensteuerprogramm diese Forderung. Weiterhin können logische Fehler, die erst an der Maschine beim Testlauf oder über einen speziellen, zum Interpreter gehörenden Debugger gefunden werden, auf dem Entwicklungsrechner in der Tabelle korrigiert werden. Da die Steuerlisten einfach austauschbar sind, ist die Übernahme auf die Maschine schnell und sicher möglich. Die kompakte Darstellung und die inhaltliche Überschaubarkeit bilden einen hohen Dokumentationswert, so daß auch nach längerer Pause gezielte Änderungen im Programm möglich sind.
Für die Verwendung einer Steuerungssoftware ist stets das Preis-Leistungs-Verhältnis heranzuziehen. Voraussetzung ist dabei erst einmal die Lösbarkeit der Aufgabe mit der gewählten Software. Entscheidend für die ökonomischen Gesichtspunkte ist dann der Aufwand von der Aufgabenstellung bis zum fertigen Projekt. Hier bietet das kurz vorgestellte, modulare Tabellensteuerprogramm sehr gute Möglichkeiten. Die Programmformulierung in Tabellen mit der Wenn-dann-Grundform kommt der Ausbildung des Maschinenkonstrukteurs wesentlich näher als zum Beispiel der Kontaktplan oder die Anweisungsliste. Auch kurze Programmentwicklungszeiten und die kompakte Dokumentation sind gegeben. Schließlich wird, sowohl im Entstehungsprozeß der Software als auch an der Maschine, der Test der Programme gut unterstützt. Es sind dazu keine gesonderten Anwenderleistungen erforderlich.

✉ KONTAKT ☏
Mikroelektronik/Software, Eugen-Fritsch-Straße 6, O-9900 Plauen; Tel. 2 27 78

Prüfung der Laufwerke

Michael Eiker

Jeder Programmierer wird bestrebt sein, seine Applikation soweit wie eben möglich „wasserdicht" zu programmieren.
Auftretende Bedienungs- oder Systemfehler sollten so abgefangen werden, daß eine sinnvolle Fehlerbehandlung stattfindet und Meldungen wie „PRINTER NOT READY... RETRY/ABORT?", nach denen man dann wieder am DOS-Prompt landet, vermieden werden. (hk)

Die meisten dieser Fehler lassen sich sehr elegant über die ERRORSYS.PRG von Clipper abfangen, einige Überprüfungen unterbleiben jedoch leider nach wie vor. Man ist als Programmierer gehalten, diese möglichen Fehlerquellen selber abzufangen und entsprechend zu behandeln.
Zu dieser Art von Fehlern gehören u. a. auch die Laufwerksabfragen, also die Überprüfung, ob das Laufwerk bereit ist, die Diskette formatiert ist usw.
Nun, diese Fehler einzeln zu überprüfen ist mit den reinen Clipper-Kommandos und Funktionen, ohne Zuhilfenahme fremder Tools, schwer zu realisieren. Es besteht jedoch die Möglichkeit, mittels der sog. Low-Level-Funktionen von Clipper eine Laufwerksprüfung in der Form vorzunehmen, daß überprüft wird, ob das Schreiben grundsätzlich möglich ist. Ist dies der Fall, dann läßt sich die Diskette im Normalfall auch einwandfrei lesen.
Interessant ist die folgende Routine, in der eine solche Abprüfung für Clipper-Anwender realisiert ist, die nur mit Clipper, also ohne Toolboxen arbeiten. Wichtig für das Testen eines Laufwerkes ist primär seine generelle Beschreibbarkeit. Festzustellen, ob eine Diskette eingelegt, ob diese beschreibbar oder ob das Laufwerk geschlossen ist, das ist sicherlich ganz angenehm und kann ein Programm anwenderfreundlicher machen, aber unbedingt notwendig ist es sicherlich nicht.
Nun könnte man natürlich hingehen und einfach versuchen, eine Testdatei auf das zu prüfende Laufwerk zu kopieren. Wenn man dies mit der normalen COPY-Routine aus Clipper oder DOS versucht, bekommt man sofort, ohne diese abfangen zu können, eine DOS-Fehlermeldung präsentiert. Genau das will man ja verhindern.
Die Frage ist also, ob es eine Möglichkeit gibt, versuchsweise eine Datei zu erzeugen, ohne daß DOS eingeschaltet wird, wenn's mal danebengeht. Um es kurz zu machen, es gibt diese, und zwar in Form der Low-Level-Funktion FCREATE().
Diese Funktion gibt im Falle eines Scheibfehlers auf ein nicht bereitstehendes Laufwerk eine -1 (numerisch) zurück. Eine Fehlermeldung im eigentlichen Sinne erfolgt nicht, sondern ein Fehler äußert sich eben nur in der Rückgabe des Wertes -1. Das ist genau das, worauf es ankommt.

```
*************************************************************
* Programm :    CHKDISK.PRG                                 *
* Autor    :    Michael Eiker                               *
* Bemerkung:    Über die Funktion CHK_DSK() läßt sich ein   *
*                beliebiges DOS-Laufwerk auf Bereitschaft   *
*                abprüfen. Es werden nur Clipper-Funktio-   *
*                nen und -Kommandos verwendet               *
*************************************************************

* Testprogramm zur Demonstration der Funktion CHK_DSK()
*
CLEAR
laufwerk = "A"
DO WHILE laufwerk <> " "
 @ 10,10 SAY "Zu prüfendes Laufwerk:
                              " GET laufwerk PICTURE "!"
 READ
 IF .NOT. EMPTY(laufwerk)
   @ 12,10 SAY "Laufwerk " + laufwerk + " bereit ? :"
   @ 32,10 SAY CHK_DSK(laufwerk)
 ENDIF
ENDDO
CLEAR
RETURN

*********************************************
* Funktion  :   CHK_DSK()                   *
* Autor     :.  Michael Eiker               *
* Bemerkung :   Prüft beliebiges DOS-       *
*                 Laufwerk auf Bereitschaft *
*********************************************
FUNCTION chk_dsk
 PARAMETERS lwname
 checkdat = lwname + ":CHK.$$$"    && Laufwerk und Dateiname
 ok_check = FCREATE(checkdat,0) && Versuch Datei anzulegen
 IF ok_check <> -1
   ERASE &checkdat                       && Checkfile löschen
 ENDIF
RETURN (ok_check <> -1)   && Rückgabe .T. wenn erfolgreich
*
* EOF: CHKDISK.PRG
*****************
```

Listing zur Funktion CHKDSK(), eingebunden in ein kurzes Testprogramm

In der Funktion CHK_DSK() ist die komplette Laufwerksabfrage realisiert. CHK_DSK() bekommt als Parameter den Laufwerksbezeichner, also z. B. „A“ als ein Zeichen langen String übergeben. In Kombination mit dem Namen einer Testdatei, wir haben in unserem Beispiel „CHK.$$$“ gewählt, versucht die Funktion FCREATE() nun, diese Datei auf dem angegebenen Laufwerk anzulegen.
Ist diese Anlage erfolgreich, wird sie über ERASE wieder gelöscht, und die Funktion gibt logisch .T. (True) zurück. Ist beim Schreiben ein Fehler aufgetreten, so liefert FCREATE() den Wert -1 und logisch .F. (False).
Es ist also auch ohne Zusatzlibraries möglich, eine einfache Laufwerksabfrage schon mit acht Zeilen Quellcode (der Funktion CHK_DSK() zu realisieren, wobei sich diese Funktion noch weiter optimieren ließe. Wir haben das in unserem Beispiel unterlassen, weil dies doch ein wenig auf Kosten der Übersichtlichkeit gegangen wäre.
Wem diese Routine jedoch nicht ausreichen sollte, der sei auf die Nantucket Tools II verwiesen, die sehr umfassende Funktionen zur Laufwerksprüfung enthalten. Über diese Funktionen können Sie z. B. prüfen, ob eine Diskette fehlerhafte Sektoren enthält, Sie können den freien Speicherplatz bzw. die Datenbelegung feststellen, aber auch eine Überprüfung auf Schreibschutz ist jederzeit möglich.

Nachdruck aus Computer Persönlich, Heft 17/90

MS-DOS-Tip

Dateien mit Rückmeldung löschen

Nino Turiansky

Die residenten MS-DOS-Befehle zum Löschen von Dateien Del und Erase haben den Nachteil, daß die Namen der wirklich gelöschten Dateien nicht angezeigt werden. Das ist aber oft, besonders bei Verwendung von Jokerzeichen, erwünscht, um zu sehen, ob nicht versehentlich Dateien gelöscht oder mit Schreibschutz versehene Dateien nicht gelöscht wurden.

Ich möchte Ihnen im folgenden ein kurzes Assemblerprogramm vorstellen, das diesen Nachteil beseitigt (Bild 1). Es benötigt weniger als 1 KByte Speicherplatz und bringt damit die Vorteile der Assemblersprache zur Geltung, die angesichts der Möglichkeiten von Hochsprachen ein wenig ins Abseits geraten ist.

Das Programm beginnt an der Marke Start und lädt als erstes die Kommandostringlänge aus dem Programm-Segment-Präfix (PSP), die sich an der Adresse 80H befindet. Wurde kein Dateiname übergeben, so wird das Programm mit der Ausgabe einer Syntaxhilfe abgebrochen.

Der Dateiname, der auch eine Laufwerksangabe und Jokerzeichen enthalten darf, ist im ersten File Control Block (FCB) ab Adresse 5CH des PSP gespeichert. Ein Pfadname darf allerdings nicht mit übergeben werden, da dann der FCB anstelle des Dateinamens Leerzeichen enthält. Das wird ebenfalls über-

```
cr          equ     0dh
lf          equ     0ah
space       equ     ' '
beep        equ     07h
dos         equ     21h
findfirst   equ     11h
findnext    equ     12h
delete      equ     13h

code     segment 'CODE'

assume  cs:code, ds:code, es:code,ss:stack

stack    segment stack
         dw       20 dup(?)
stack ends

synt          db ' Aufruf <DELE> [LW:]'
              db '<dateiname/-gruppe>',
              db beep,cr,lf,'$'
deleted       db ' geloescht',cr,lf,'$'
notdeleted    db ' nicht geloescht (R/O)',
              db beep,cr,lf,'$'
punkt         db '.$'
fehler        db ' Ungueltiger Dateiname.'
              db ' Keine Pfadangabe !',
              db cr,lf,'$'

dta             db          128 dup(?)
fcb             db          0,'???????????'
                db          25 dup(?)

       jmp start

outname proc
;Ausgabe Dateiname von dta
       mov dx,di
       repne scasb
       mov byte ptr [di-1],'$'
       call ausgabe
       ret
outname endp

erase  proc
;loeschen der Datei
       mov dx,offset dta
       mov ah,delete
       int dos
;Rueckkehrcode sichern
       push ax
;erstes Leerzeichen im Dateinamen
;suchen und '$' einfuegen
       mov al,' '
       mov di,offset dta+1
       mov cx,9
;Dateiname ausgeben
       call outname
;Punkt ausgeben
       mov dx,offset punkt
       call ausgabe


;erstes Leerzeichen nach Dateierweiterung
;suchen,'$' einfuegen
       mov di,offset dta+9
       mov cx,4
;Erweiterung ausgeben
       call outname
;Rueckkehrcode auswerten
       pop ax
       or  al,al
       jz  inordnung
;Fehler
       mov dx,offset notdeleted
       call ausgabe
       jmp ende
inordnung:
       mov dx,offset deleted
       call ausgabe
ende:
       ret
erase  endp

ausgabe proc
       mov ah,09h
       int dos
       ret
ausgabe endp


 ;Kommandostringlaenge testen
start:
       mov si,80h
       mov al,[si]
       cmp al,1
;nichts uebergeben
       jbe syntax
;Test auf gueltigen Dateinamen
       mov di,5dh
       mov cx,11
       mov al,space
       repe scasb
       cmp cx,1
       jb fehl
;ES initialisieren
       mov ax,code
       mov es,ax
;uebergebenen Dateinamen in fcb bringen
       mov si,5ch
       lea di,es:fcb
       mov cx,12
       rep movsb
;DS initialisieren
       mov ds,ax
;neue dta Adresse setzen
       mov dx,offset dta
       mov ah,1ah
       int dos
;ersten Eintrag suchen
       mov ah,findfirst
loop:
       mov dx,offset fcb
       int dos
       or al,al
;keine Datei gefunden - Ende
       jnz finished
       call erase
;naechsten Eintrag suchen
       mov ah,findnext
       jmp loop
finished:
;Programm beenden
       mov ah,4ch
       int dos
fehl:
       mov ax,code
       mov ds,ax
       mov dx,offset fehler
       call ausgabe
;kleine Syntaxhilfe
syntax:
       mov ax,code
       mov ds,ax
       mov dx,offset synt
       call ausgabe
       jmp finished

code   ends
       end start
```

***Bild 1** Das Programm Dele zeigt die Dateien an, die durch sein Aufrufen gelöscht wurden.*

prüft und im Fehlerfall eine entsprechende Meldung ausgegeben. Ansonsten wird der Dateiname in den eigenen FCB übertragen. Danach wird die DTA-Adresse auf die Marke DTA gesetzt. Dafür wird die Funktion 1AH des Interrupt 21 H benutzt. Nun kann die Suche nach dem ersten Eintrag im Verzeichnis erfolgen. Die Funktionen 11H (Finden des ersten Eintrags), 12H (Finden des nächsten Eintrags) und 13H (Datei löschen) des Interrupt 21H liefern einen Rückkehrkode im Register AL. Dieser ist 0 bei erfolgreicher Suche bzw. 0FFH, wenn ein Fehler aufgetreten ist.
Solange jetzt gültige Verzeichniseinträge gefunden werden, wird die Prozedur Erase aufgerufen. Diese benutzt die Funktion 13H zum Löschen einer Datei. Sie erlaubt keine Pfadangabe im Dateinamen. Das hat zur Folge, daß erst das entsprechende Verzeichnis eingestellt werden muß, bevor gelöscht werden kann. Auch hier wird der Rückkehrcode ausgewertet und eine Meldung über Erfolg oder Mißerfolg mit den Prozeduren Outname und Ausgabe ausgegeben. Der Dateiname wird aus dem eigenen DTA-Bereich geholt und vollständig ausgegeben.
Der Quelltext kann mit einem Texteditor eingegeben und dann durch die Befehle **MASM DELE;** und **LINK DELE;** zum fertigen Programm verarbeitet werden.

Trigonometrische Funktionen in dBase III

Michael Kretzschmar, Leipzig

Im Datenbanksystem dBase III (Plus) sind die trigonometrischen Funktionen nicht implementiert. Da man für bestimmte Berechnungen diese Funktionen benötigt, gibt es im Datenbanksystem drei Möglichkeiten, darauf zuzugreifen:
① Die trigonometrischen Funktionen werden in vorgegebener Schrittweite als Dateien eingespeichert.
② Die Nutzung von Funktionen anderer Programmiersprachen.
③ Es werden entsprechende Näherungsformeln programmiert.
Im vorliegenden Beitrag möchte ich Ihnen ein Programm für die dritte Version vorstellen, da die Rechenzeit auch bei Näherungsformeln relativ gering ist.
Da in dBase III Plus die Arbeit mit Unterprogrammen möglich ist, können die trigonometrischen Funktionen von jedem Programm genutzt werden.
Die Unterprogramme für Sinus, Kosinus, Tangens und Kotangens

```
* PROCTEST.PRG = Test Procedure-File
set talk off
public alpha,alphah,sin,cos,tan,cot
clear
sin=0.000000
cos=0.000000
tan=0.000000
cot=0.000000
input 'Winkel ALPHA in Altgrad = ' to alpha
alphah=alpha
set procedure to b:trigofun
do sinus with alpha,sin
do cosinus with alpha,cos
do tangens with sin,cos,tan
do cotan with sin,cos,cot

? 'sin( '+str(alphah,10,2)+' ) = '+str(sin,15,6)
? 'cos( '+str(alphah,10,2)+' ) = '+str(cos,15,6)
? 'tan( '+str(alphah,10,2)+' ) = '+str(tan,15,6)
? 'cot( '+str(alphah,10,2)+' ) = '+str(cot,15,6)
?
?

wait
?
accept 'Ausdruck ? (j/n)  :  ' to dr
if upper(dr)='J'
set console off
set print on
? 'sin( '+str(alphah,10,2)+' ) = '+str(sin,15,6)
? 'cos( '+str(alphah,10,2)+' ) = '+str(cos,15,6)
? 'tan( '+str(alphah,10,2)+' ) = '+str(tan,15,6)
? 'cot( '+str(alphah,10,2)+' ) = '+str(cot,15,6)
?
?
?
set print off
set console on
endif
close procedure
set talk on
return
```

Bild 1

```
****** TRIGOFUN.PRG = Trigonometrische Funktionen

procedure sinus
parameters alpha,sin

if alpha<0
alpha=-alpha
endif

n=int(alpha/360)
alpha1=alpha-n*360

do case

case alpha1<=90
alphar=alpha1

case alpha1<=180
alphar=180-alpha1

case alpha1<=270
alphar=-(alpha1-180)

case alpha1<=360
alphar=-(360-alpha1)

endcase

x=3.141593*alphar/180.000
x2=x*x
x3=x2*x
x5=x3*x2
x7=x5*x2
x9=x7*x2
x11=x9*x2
sin=x-x3/6+x5/120-x7/5040+x9/362880-x11/39916800
v=sin*sin
v=sqrt(v)
if v<=0.000001
sin=0.000000
endif

if alphah<0
sin=-sin
endif
*** Ende SINUS

*******

procedure cosinus
parameters alpha,cos

if alpha<0
alpha=-alpha
endif

n=int(alpha/360)
alpha1=alpha-n*360

do case
```

Bild 2

```
case alpha1<=90
alphar=alpha1

case alpha1<=180
alphar=-(180-alpha1)

case alpha1<=270
alphar=-(alpha1-180)

case alpha1<=360
alphar=360-alpha1

endcase

x=3.141593*alphar/180.000
x2=x*x
x4=x2*x2
x6=x4*x2
x8=x6*x2
x10=x8*x2
cos=1-x2/2+x4/24-x6/720+x8/40320-x10/3628800

if (alpha1>=90 .and. alpha1<=270)
cos=-cos
endif

v=cos*cos
v=sqrt(v)
if v<=0.000001
cos=0.000000
endif

*** Ende COSINUS

********

procedure tangens
parameters sin,cos,tan
if cos=0.000000
tan=100000000000000000
else
tan=sin/cos
endif

*** Ende TANGENS

********

procedure cotan
parameters sin,cos,cot
if sin=0.000000
cot=100000000000000000
else
cot=cos/sin
endif

*** Ende COTANGENS

********

return
```

sind in der Prozedur-Datei Trigofun.Prg gespeichert (Bild 2). Für Sinus und Kosinus werden Taylorreihenentwicklungen mit einer vorgegebenen Genauigkeit verwendet.
Die Abarbeitung gestattet eine Eingabe von jedem beliebigen Winkel in Altgrad.
Bei Neugrad wäre nur ein Austausch bei der Berechnung des Bogenmaßes (Division durch 200 statt durch 180) erforderlich.
Die eingegebenen Winkel werden grundsätzlich auf Winkel zwischen 0 und 90 Altgrad reduziert, da die Taylorreihen nur in diesem Bereich exakt konvergieren.
Die Werte für *unendlich* bei Tangens und Kotangens werden mit einer sehr großen Zahl dargestellt.
Damit ergibt sich in der Bildschirm- bzw. Druckerausgabe ein Formatüberlauf, der mit Sternen angezeigt bzw. gedruckt wird. Die Prozeduren werden vom Hauptprogramm Proctest.Prg (Bild 1) aufgerufen. Die Ergebniswerte können wahlweise auch gedruckt werden.

Die Interrupts 25H und 26H unter Turbo-Pascal

Christian Hanisch, Berlin

Das direkte sektororientierte Lesen und Schreiben von Daten auf einen externen Datenträger kann unter MS-DOS auf zweierlei Weise erfolgen: Mit Hilfe des BIOS-Interrupts 13H oder mit den Betriebssysteminterrupts 25H und 26H. Um den BIOS-Interrupt 13H nutzen zu können, sind eine Reihe Parameter erforderlich. So muß beispielsweise beim Lesen von Sektoren von der Festplatte angegeben werden, mit welchem Kopf und in welcher Spur gelesen werden soll. Gleiches gilt für das Schreiben von Sektoren. Da man jedoch häufiger die logische Sektornummer, die das Betriebssystem verwendet, als Zugriffsparameter zur Verfügung hat, bieten sich die Betriebssysteminterrupts 25H und 26H an, die anhand dieser logischen Sektornummer den physischen Sektor auffinden können.
Interrupts werden in Turbo-Pascal prinzipiell mit Hilfe der Prozedur Intr aufgerufen. Beim BIOS-Interrupt 13H ist dies auch praktikabel. Hingegen versagt es bei den Interrupts 25H und 26H. Der Grund ist darin zu sehen, daß diese Betriebssysteminterrupts das Flag-Word auf dem Interruptstapel belassen. Der Programmierer muß selbst dafür sorgen, daß dieses Word dort entfernt wird.
Um die Interrupts 25H und 26H in Turbo-Pascal verwenden zu können, bieten sich deshalb Inline-Makros an, wie sie im Bild 1 zu sehen sind. Mit Hilfe dieser Makros ist das absolute Lesen und Schreiben von Daten auf Datenträgern möglich. Eine denkbare praktische Anwendung – in welchem Zusammenhang auch immer – ist das im Beispiel gezeigte Lesen und Schreiben des BOOT-Sektors.

```
 1: PROGRAM INT25_26_makro; (* Turbo Pascal 5.x *)
 2: USES dos;
 3: VAR bootsec : ARRAY[1..512] OF BYTE;
 4:     i       : INTEGER;
 5:
 6: PROCEDURE intr_25(Data_Seg,Data_Ofs,First_Sect,Count_Sect,Drive_Nr:word);
 7: { Data_Seg    : Segment des Pufferbereiches, in dem die gelesenen
 8:                 Sektoren abgelegt werden
 9:   Data_Ofs    : Offset des Pufferbereiches, in dem die gelesenen
10:                 Sektoren abgelegt werden
11:   First_Sect  : logische Sektornummer, des ersten zu lesenden
12:                 Sektors
13:   Count_Sect  : Anzahl der zu lesenden Sektoren
14:   Drive_Nr    : Laufwerks-Nr. 0 - A, 1 - B, 2 - C, ...          }
15:
16: INLINE($58          { pop ax        ; Laufwerk                     }
17:       /$59          { pop cx        ; Anzahl Sectoren              }
18:       /$5a          { pop dx        ; Beginn mit Sector            }
19:       /$5b          { pop bx        ; Data_ofs nach BX             }
20:       /$5f          { pop di        ; Data_Seg zwischenspeichern   }
21:       /$1e          { push ds       ; Turbo Datensegment retten    }
22:       /$55          { push bp       ; BP retten                    }
23:       /$8e/$df      { mov ds,di     ; Data_Seg nach DS             }
24:       /$cd/$25      { INT 25H       ;                              }
25:       /$9d          { popf          ; Flags zurück                 }
26:       /$5d          { pop bp        ; BP wiederherstellen          }
27:       /$1f);        { pop ds        ; Turbo DSEG wiederherstellen  }
28:
29: PROCEDURE intr_26(Data_Seg,Data_Ofs,First_Sect,Count_Sect,Drive_Nr:word);
30: INLINE($58          { pop ax        ; Laufwerk                     }
31:       /$59          { pop cx        ; Anzahl Sectoren              }
32:       /$5a          { pop dx        ; Beginn mit Sector            }
33:       /$5b          { pop bx        ; Data-Ofs nach BX             }
34:       /$5f          { pop di        ; Data_Seg zwischenspeichern   }
35:       /$1e          { push ds       ; Turbo Datensegment retten    }
36:       /$55          { push bp       ; BP retten                    }
37:       /$8e/$df      { mov ds,di     ; Data_Seg nach DS             }
38:       /$cd/$26      { INT 26H       ;                              }
39:       /$9d          { popf          ; Flags zurück                 }
40:       /$5d          { pop bp        ; BP wiederherstellen          }
41:       /$1f);        { pop ds        ; Turbo DSEG wiederherstellen  }
42:
43: BEGIN
44:   (* Lesen des Boot-Sectors: *)
45:   intr_25(seg(bootsec),ofs(bootsec),0,1,2);
46:   (* Ausschreiben der OEM-Kennung: *)
47:   FOR i:=1 TO 8 DO write(CHAR(bootsec[3+i]));
48:   writeln;
49: END.
```

***Bild 1** Inline-Makros zum Aufruf des Betriebssysteminterrupts 25H und 26H*

Semantische Netze am PC entwickeln

Stephan Weinmeister, Dresden

In diesem Artikel wird ein Programm vorgestellt, das es erlaubt, semantische Netze /1/, /2/, /3/ von einem oder mehreren Bearbeitern rechnergestützt erzeugen zu lassen. Die Anwendung dieses Programms ist überall dort vorteilhaft, wo der Umfang der semantischen Netze nicht mehr mit Stift und Papier zu bewältigen ist oder wo mehrere Fachleute an der Entwicklung dieser Netze beteiligt sind.

Vor dem Aufbau der Wissensbasis für ein Expertensystem ist es sinnvoll, das zugrunde zu legende Wissensgebiet abzustecken. Dies geschieht in grober Form durch Absprache zwischen Systementwicklern und Experten. Für die detaillierte Beschreibung haben sich semantische Netze bewährt.
Mit Hilfe dieser Netze kann der interessierende Teil des Fachgebietes aufbereitet werden, ohne daß dadurch die später verwendete Wissensrepräsentation des Expertensystems bereits festgelegt wäre. Die Aufgabe des semantischen Netzes besteht „nur" darin, einen Überblick über das in der Wissensbasis abzulegende Wissen zu geben. Das geschieht in Form von Objekten (Konzepten, Knoten), die durch Relationen (Beziehungen, Kanten) miteinander verbunden sind. Dies ist eine einfache, unkomplizierte und übersichtliche Methode, wenn das Netz klein ist oder wenn es rechnergestützt entwickelt und betrachtet werden kann.
Soll ein Expertensystem aufgebaut werden, dessen Wissen durch semantische Netze repräsentiert und in der Wissensbasis abgelegt wird, dann kann das unten beschriebene Programm auch zum Wissenserwerb des Systems benutzt werden. Dies ist beispielsweise bei Diagnose-Expertensystemen der Fall, deren Wissensbasis im Prinzip durch einen Fehlersuchbaum gebildet wird. Ein solcher Fehlersuchbaum verbindet Aussagen über mögliche Defekte (die Objekte) durch Fragestrategien (die Relationen). Er stellt damit ein spezielles semantisches Netz dar.
Das Entwerfen eines semantischen Netzes kann als ein zyklischer Vorgang aufgefaßt werden. Mit einem Startobjekt beginnend, werden anschließend eine dazugehörige Relation und das Objekt, zu dem sie führt, benannt. Der weitere Ausbau des Netzes besteht nun darin, jeweils neue Relationen und dazugehörige Objekte einzuführen. Dabei kann ein beliebiges, bereits existierendes Objekt als neuer Ausgangspunkt dienen.
Der beschriebene zyklische Vorgang wird vom Programm Semanetz organisiert. Nach der Eingabe des Startobjektes wird dem Nutzer ein Menü mit Standardrelationen angeboten. (Diese sind im Programmtext festgelegt und können vom Nutzer verändert werden.) Wird eine der angebotenen Relationen ausgewählt, erscheint ein neues Menü, das alle Objekte auflistet, zu denen diese Relation führt. Diese Liste kann durch den Nutzer bei der Bewegung durch das Netz (durch die Menüs) jederzeit verändert werden (Bild 1). Wenn noch kein Objekt gespeichert ist, erscheint die Aufforderung, die zur Relation gehörenden Objekte einzugeben.

Zur Unterstützung unerfahrener Nutzer verfügt das Programm Semanetz über umfangreiche On-line-Hilfe- und Beispielinformationen. Dabei stehen Hilfsinformationen sowohl für die Steuerung des Programms (Tastaturfunktionen) als auch zum Inhalt von Eingaben zur Verfügung.
Außerdem bietet das Programm die Möglichkeit, (mittels eines einfachen Editors) zu jedem Objekt auch verschiedene Texte einzugeben. Das kann zum Beispiel genutzt werden, um sich auf das Objekt beziehende Regeln oder Gesetzmäßigkeiten einzugeben oder Parameter und andere Einflußfaktoren festzuhalten oder um Literaturstellen zu speichern, die sich auf die Relation und das Objekt beziehen.
Der Vorteil für die gemeinsame Arbeit mehrerer Fachleute an einem semantischen Netz wird hierbei bereits deutlich. Jeder Nutzer sieht bei der Arbeit mit dem Programm Semanetz, was von allen vorhergehenden Nutzern eingegeben wurde. Er kann dann das Netz ausbauen, indem er vorhandene Objektlisten erweitert oder neue Objekte einfügt.

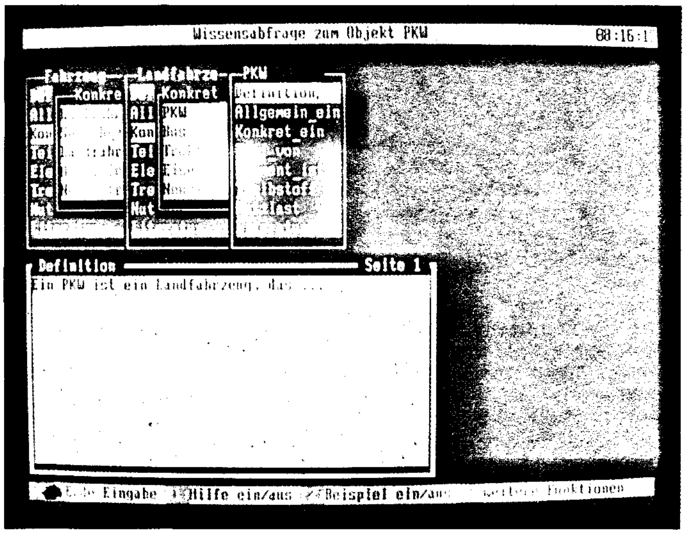


Bild 1 Beispiel für Menügestaltung im Programm Semanetz

Derzeitig ist es mit dem Programm noch nicht möglich, Ausschnitte des Netzes grafisch darzustellen. Dadurch wäre eine vereinfachte Orientierung für den Nutzer gegeben. Ebenso sollen auch Möglichkeiten für Suchanfragen des Nutzers (z. B. nach einem bestimmten Objekt) künftig realisiert werden. Geplant ist ebenfalls, durch geringfügige Veränderungen am Programm, die Eingabe von Objekten als *Frames* zu unterstützen (vgl. /1/, /4/, /5/).
Langfristig soll Semanetz zu einer Expertensystem-Komponente entwickelt werden, die es ähnlich wie bei /6/ erlaubt, daß der Experte sein Wissen direkt in das System eingeben kann. Dazu sind Beschreibungs- und Darstellungsmittel an der Benutzeroberfläche notwendig, die der Fachmann ohne tiefgründige Informatikkenntnisse verstehen und benutzen kann. Die Umwandlung in die interne Wissensdarstellung des Expertensystems erfolgt dann ebenfalls durch diese Wissenserwerbskomponente.
Das Programm Semanetz ist in Modula-2 geschrieben. Es wurde mit TopSpeed-Modula V 1.15 unter MS-DOS auf dem A 7150 und dem EC 1834 entwickelt.

Literatur

/1/ Dörne, H.: Wissensrepräsentation – Ideen, Aspekte, Formalismen. – In: Grundlagen der Künstlichen Intelligenz / Grabowski, J.; Jantke, K. P.; Thiele, H. (Hrsg.). – Akademie-Verlag: Berlin 1989, S. 21

/2/ Smith, J. M.; Smith, D. C. P.: Principles of Database Conceptual Design. – In: Data Base Design Techniques I: Requirements and Logical Structures (Proceedings of the NYU Symposium, New York, 1978) / Yao, S. B.; Navathe, S. B.; Weldon, J.-L; Kunii, T. L. (eds.). – Springer-Verlag: Berlin, Heidelberg, New York 1982, S. 114

/3/ Brodie, M. L.: Association: A Database Abstraction for Semantic Modelling. – In: Entity-Relationship Approach to Information Modeling and Analysis (Proceedings of the Second International Conference on E-R Approach, Washington, D. C., 1981) / Chen, P. P. (ed.). – Elsevier Science Publishers B. V.: Amsterdam, New York, Oxford 1983, S. 577–602

/4/ Neill, G.: Künstliche Intelligenz: Wie Sie Ihren Computer zum Denken bringen. – (Übersetzung von: Artificial Intelligence – Making machines „think". – TAB BOOKS Inc.: Blue Ridgge Summit, Penna., 1979). – Luther-Verlag: Sprendlingen 1983

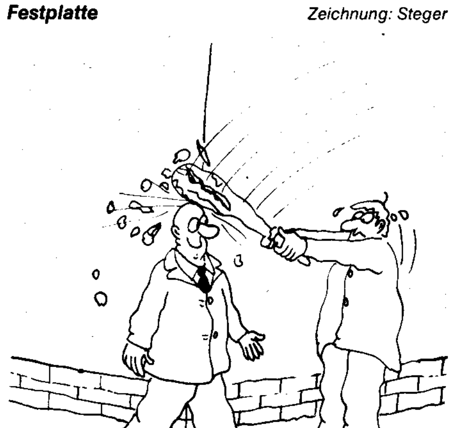
/5/ Stede, M.: Einführung in die künstliche Intelligenz. – Luther-Verlag: Sprendlingen 1983

/6/ Rieckert, W. F. J.: Werkzeuge und Systeme zur Unterstützung des Erwerbs und der objektorientierten Modellierung von Wissen. Institut für Informatik der Universität Stuttgart, Diss. A., Stuttgart 1986

✉ KONTAKT ☏

Technische Universität Dresden, Sektion Informationstechnik/B5, Mommsenstraße 13, O-8027 Dresden; Tel. 463 53 99

Festplatte *Zeichnung: Steger*



MS-DOS-Tip

Bildschirminhalte abspeichern

Wolfgang Neumann, Berlin

Sicherlich standen Sie auch schon einmal vor dem Problem, ein Programm dokumentieren zu müssen. Neben allgemeinen Informationen sollte die Dokumentation auch eine ausführliche Erläuterung der Bedienerführung enthalten. Ein großer Teil der dazu nötigen Informationen (Menüs, Eingabemasken, Hilfsausschriften usw.) wird während des Programmlaufes auf dem Bildschirm geliefert. Wie kann man nun diese Informationen in sein Textsystem übernehmen?

Eine Möglichkeit besteht in der Manipulation des zu dokumentierenden Programms (Einfügen einer speziellen Routine zur Bildschirm-Speicherung in die Tastatur-Schleife). Das erfordert aber einen Eingriff in den Quelltext und setzt damit spezielle Programmierkenntnisse voraus.

Wollen Sie dies umgehen, können Sie eigentlich nur noch mit Hilfe der Print-screen-Taste den Bildschirminhalt ausdrucken und ihn anschließend mühsam wieder eintippen.

Eine wesentliche Vereinfachung wäre, die Print-screen-Taste zu nutzen, um den Bildschirminhalt statt auf den Drucker in eine Textdatei zu schreiben.

Die Umleitung des Print-screen-Interrupts

Durch die Print-screen-Taste wird der BIOS-Interrupt 05 aktiviert. Er kann von einem speicherresidenten Programm auf eine eigene Routine umgeleitet werden, die dann anstelle des Drucks den Bildschirminhalt in eine Textdatei ausgibt.

Da die Adressen des Bildwiederholspeichers und dessen Aufbau bekannt sind, ist es lediglich erforderlich, den Bildwiederholspeicher zu lesen und in einer Textdatei zu speichern. Das Abspeichern ist allerdings etwas problematisch. Wenn Sie sich kein eigenes Dateisystem schreiben wollen, müssen Sie die entsprechenden DOS-Rufe für das Erzeugen und das Füllen der Textdatei nutzen. Dabei ist zu beachten, daß eine Interrupt-Routine, die einen DOS-Ruf unterbricht, ihrerseits keine DOS-Rufe enthalten darf. Da die Print-screen-Taste zu jedem beliebigen Zeitpunkt gedrückt werden kann und die meisten Programme laufend DOS-Rufe nutzen, würde eine solche Interrupt-Routine nach ihrer Aktivierung in der Regel abstürzen.

Eine einfache Lösung dieses Problems liegt in der Aufteilung der beiden Funktionen auf zwei verschiedene Programme:

① Einlesen von Bildschirminhalten in einen reservierten RAM-Bereich

② Abspeichern der eingelesenen Informationen auf eine Textdatei.

Das Programm Scsave verbleibt resident im Hauptspeicher und leitet den Interrupt 05 auf eine Routine um, die maximal 32 Bildschirminhalte im RAM ablegt. Dazu sind keine DOS-Rufe nötig. Das Programm Scget wird nach dem Einlesen der Bildschirminhalte und nach Beendigung des zu dokumentierenden Programms gestartet und speichert den RAM-Inhalt in eine Textdatei ab. Hier können durchaus DOS-Rufe verwendet werden. Bei diesem Konzept ist nur noch die Kommunikation zwischen beiden Programmen zu gewährleisten, das heißt die Übergabe der Adresse des Puffers für die Bildschirminhalte. Hierfür kann ein Teil des Adreßbereiches der Interruptvektoren (0000H bis 0400H) benutzt werden. Er enthält die Anfangsadressen der Interrupt-Behandlungsroutinen, die beim Systemstart vom Betriebssystem eingetragen werden. Dabei wird ein bestimmter Bereich für Anwenderprogramme freigehalten (Adresse: 0180H bis 01A0H = Nr. 60H bis 80H).

Von einem Anwenderprogramm können beliebige Adressen in diesen Bereich eingetragen werden, also auch die Anfangsadresse eines Speicherbereiches. Diese Informationen bleiben auch nach der Beendigung des eintragenden Programms erhalten und können von anderen Programmen abgefragt werden.

Die Bilder 1 und 2 zeigen die für Turbo-Pascal ab der Version 4.0 geschriebenen Programme Scsave und Scget.

Durch die Nutzung der komfortablen Standardprozeduren für die Interrupt-Behandlung ergibt sich ein besonders kurzer Quelltext.

Wenn das residente Programm Scsave weniger Speicherplatz belegen soll, kann die Konstante BildZahl auf minimal 1 verringert werden.

Wird von dem zu dokumentierenden Programm der Print-screen-Interrupt ebenfalls umgeleitet, so kann die Konstante SaveInt auf 19H gesetzt werden. Der Aufruf erfolgt dann mit der Tastenkombination <Ctrl>-<Alt>-<Del> (ursprünglich Systemstart).

Wird zufällig der Interrupt 60H von einem anderen residenten oder von dem zu dokumentierenden Programm genutzt, kann die Konstante BufInt auch auf einen anderen freien Interrupt-Vektor gesetzt werden.

Änderungen dieser Konstanten müssen aber unbedingt in beiden Programmen gleichermaßen vorgenommen werden.

Die Arbeit mit den Programmen Scsave und Scget

Nachdem das Programm Scsave und das zu dokumentierende Programm aufgerufen wurden, kann durch das Betätigen der Print-screen-Taste an maximal 32 interessanten Stellen der Bildschirminhalt gespeichert werden; die Nummer des Bildes wird in der linken oberen Bildschirmecke angezeigt. Ist das zu dokumentierende Programm beendet, können durch den Aufruf des Programms Scget die bisher nur im RAM gesammelten Darstellungen endgültig auf der Diskette

```
PROGRAM SCSAVE;
USES dos;

{$M 1024,0,0 }
{ Speicherplatzaufteilung: 1 KByte Stack, 0 Byte Heap }

CONST BildZahl=32;  { Max. Anzahl Bilder im Puffer }
      SaveInt=$05;  { Nr. des Aktivierungs-Interrupts }
      BufInt=$60;
      { Nr. des Interrupts zur Übergabe der Puffer-Adresse }

TYPE  BildType = array[1..25,1..80] of char;
      BufType  = record
                   Zaehler : Byte;
                   Bild    : array[1..BildZahl] of BildType;
                 end;
      ScreenType = array[1..25,1..80] of record z:char;a:Byte end;

VAR   Reg         : registers;
      Bild        : ^ScreenType;
      Buf         : BufType;


PROCEDURE SchreibBild;
interrupt;
{ schreibt ein Bildschirmbild in den Puffer }
var i,j : Byte;
    S   : string[2];
    temp:Longint;
BEGIN
   if Buf.Zaehler>=BildZahl then EXIT;
   { Abbruch, wenn Puffer voll }
   Move(Bild^,temp,4); {  linke obere Bildschirmecke retten }
   inc(Buf.Zaehler);
   for j:=1 to 80 do Buf.Bild[Buf.Zaehler,1,j]:=Bild^[1,j].z;
   { schreibt die erste Bildschirmzeile in den Puffer }
   str(Buf.Zaehler:2,S);
   for i:=1 to 2 do Move(S[i],Bild^[1,i].z,1);
   { Zähler in der linken oberen Bildschirmecke anzeigen }
   for i:=2 to 25 do for j:=1 to 80 do
       Buf.Bild[Buf.Zaehler,i,j]:=Bild^[i,j].z;
   { Schreibt 2.bis 25. Zeile in den Puffer }
   Move(Temp,Bild^,4);
   { Wiederherstellen der linken oberen Bildschirmecke }
END;

BEGIN
   intr($11,reg);
   { Schalterstellung zur Bestimmung der Bildadresse lesen }
   if (reg.ax and 24)=24 then Bild:=Ptr($B000,$0)
                         else Bild:=Ptr($B800,$0);
   { Zuweisung zur Pointer-Variablen Bild:                   }
   { Monochrom-Schirm=$B000:$0, Grafik-Schirm=$B800:$0       }
   SetIntVec(SaveInt,@SchreibBild);
   { Umleiten des Print-screen-Interrupts auf die eigene
     Routine SchreibBild                                     }
   SetIntVec(BufInt,@Buf);
   { Eintragen der Puffer-Anfangsadresse (für SCGET)         }
   Buf.Zaehler:=0;
   Keep(0);        { Programm speicherresident beenden       }
END.
```

***Bild 1** Das Programm Scsave legt bei Betätigung der Print-Screen-Taste den aktuellen Bildschirminhalt im RAM ab.*

```
PROGRAM SCGET;
USES dos;

CONST BildZahl=32;  { maximale Anzahl Bilder im Puffer }
      BufInt=$60;
      { Nummer des Interrupts zur Übergabe der Puffer-Adresse }

TYPE BildType = array[1..25,1..80] of Char;
     BufType  = record
                  Zaehler : Byte;
                  Bild    : array[1..BildZahl] of BildType;
                end;

VAR  Buf          : ^BufType;
     Temp         : Pointer absolute Buf;
     T            : Text;
     Zaehler      : Byte;

PROCEDURE HolBild(Zaehler:Byte);
{ schreibt ein Bild in die Textdatei }
var i,j : Byte;
BEGIN
   writeln(Zaehler:2);
   for i:=1 to 25 do
   begin
      for j:=1 to 80 do if Buf^.Bild[Zaehler,i,j]=^Z then
          write(T,' ')
          { <Ctrl>-<Z> als Textende-Kennzeichen ausblenden }
          else write(T,Buf^.Bild[Zaehler,i,j]);
      writeln(T);
   end;
END;

BEGIN
   assign(T,paramstr(1));
   { Aufruf-Parameter ist der Dateiname der Textdatei}
   rewrite(T);
   GetIntVec(BufInt,Temp);
   { Holt die Anfangsadresse des Puffers Buf }
   for Zaehler:=1 to Buf^.Zaehler do HolBild(Zaehler);
   close(T);
   Buf^.Zaehler:=0;    { Bild-Zähler initialisieren }
END.
```

***Bild 2** Das Programm Scget schreibt die im RAM gespeicherten Bildschirminhalte in eine anzugebende Textdatei.*

oder der Festplatte gespeichert werden. Dazu benötigt das Programm Scget beim Aufruf unbedingt den Namen der Textdatei als Parameter in der Kommandozeile. Bevor mit der entstandenen Textdatei gearbeitet werden kann, ist möglicherweise noch die Konvertierung in das spezielle Format des von Ihnen verwendeten Textsystems erforderlich. Auf diese Weise können beliebig viele Programme dokumentiert werden.

Abschließend sei darauf hingewiesen, daß die hier vorgestellten Programme nur für die alphanumerische Nutzung des Bildschirms funktionieren. Das Prinzip läßt sich aber auch für Grafiken nutzen, wenn der Grafik-Bildwiederholspeicher direkt adressierbar ist (z. B. bei CGA-Bildschirm-Treiber).

Sortiertes Verzeichnis ausgeben

Roland Radandt, Berlin

Der DOS-Befehl *Dir* listet den Verzeichnisinhalt in unsortierter Form, das heißt in der Reihenfolge der Einträge, aus. Manchmal ist es wünschenswert, das Verzeichnis nach einem bestimmten Kriterium sortiert ausgegeben zu bekommen. Verschiedene Tools wie QDOS oder der Norton Commander unterstützen das, wobei aber häufig nicht die gesamte Palette der Sortier-Möglichkeiten zur Verfügung steht. Die Batch-Datei Sdx.bat bietet den vollständigen Sortierumfang. Sortiert werden kann nach dem Dateinamen, der Erweiterung, der Größe, dem Tag, dem Monat, dem Jahr und nach der Zeit. Aufgerufen wird sdx.bat in der folgenden Form:

sdx <Parameter>

Als Parameter sind entsprechend obiger Aufzählung n, e, s, d, m, y und t zulässig, wobei aber jeweils nur einer angegeben werden kann. Ohne Parameter bzw. beim Aufruf mit Fragezeichen wird ein Hilfstext angezeigt. Wird beispielsweise *sdx t* aufgerufen, wird der Verzeichnisinhalt, aufsteigend nach der Uhrzeit, geordnet ausgegeben. Sortiert wird immer das aktuelle Verzeichnis, und die Ausgabe erfolgt auf dem Bildschirm. Für MS/PC-DOS unterhalb der Version 3.30 ist das Zeichen @ in dem Befehl @echo off zu entfernen. Leerzeilen innerhalb der Batch-Datei werden als **echo.** eingegeben.

```
@echo off
rem (c) R.Radandt -- Klinikum Berlin-Buch, Hs. 505 F
rem Sortiertes Verzeichnis ausgeben
rem Version 1.2
cls
if "" == "%1" goto expl
if ? == %1 goto expl
goto ok
:expl
echo Eingabe-Syntax für sdx (sorted directory)
echo.
echo sdx [Parameter]
echo Das aktuelle Verzeichnis wird sortiert nach dem an-
echo gegebenen Parameter angezeigt:
echo.
echo n - Name, e - Erweiterung, s - Dateigröße, d - Tag,
echo m - Monat, y - Jahr, t - Zeit
echo.
echo Es ist jeweils nur ein Parameter zulässig.
echo Beispiel: sdx e
goto end
:ok
if not "" == "%2" goto expl
for %%s in (n N e E s S d D m M y Y t T) do if %1 == %%s goto %%s
goto expl
:n
dir | sort | more
goto end
:e
dir | sort/+10 | more
goto end
:s
dir | sort/+16 | more
goto end
:d
dir | sort/+24 | more
goto end
:m
dir | sort/+27 | more
goto end
:y
dir | sort/+30 | more
goto end
:t
dir | sort/+34 | more
:end
```

8086-Tip

Segmentregister

Die Mikroprozessoren vom Typ 8086 können einen Speicherraum von 1 MByte direkt adressieren. Eine normale 16-Bit-Adresse reicht jedoch für 64 KByte aus. Die fehlenden 4 Bit werden durch das Konzept der Segmentregister erzeugt: Eine im Prozessor gespeicherte 16-Bit-Segmentadresse wird um 4 Bit nach links verschoben und zur eigentlichen Adresse, der Offset-Adresse, addiert. Der Prozessor 8086 enthält 4 Segmentregister, die dem Programmierer mehr oder weniger zur freien Verfügung stehen:

CS Codesegment **DS** Datensegment
SS Stacksegment **ES** Extrasegment

Ohne Umladen der Segmentregister kann somit gleichzeitig auf Daten in 4 verschiedenen 64-KByte-Bereichen zugegriffen werden. Die 4 Segmente dürfen sich überlappen und im unteren Grenzfall auch zusammenfallen (8080-Speichermodell). Wird die Segmentadresse um eine Einheit erhöht, springt der adressierte Bereich um 16 Byte (= 1 Paragraph).

Alle Befehle beziehen sich grundsätzlich auf das Codesegment. Deshalb ist das Ändern des CS-Registers durch einfache Ladebefehle (MOV CS, XX oder POP CS) nicht möglich. Das CS-Register kann nur durch „weite" Sprungbefehle (JMPF, CALLF, RETF, IRET) geändert werden.

Datenzugriffe können prinzipiell über alle 4 Segmentregister erfolgen. Es gibt jedoch feste oder zumindest standardmäßige Zuordnungen von Befehlen und Arbeitsregistern zu den Segmentregistern. Die genaue Kenntnis dieser Regeln ist für den Programmierer unerläßlich. Sie verhilft zudem zu einer speicher- und zeiteffektiven Programmierung.

Standardmäßig beziehen sich alle direkten Datenzugriffe (über Offsetadressen) auf das Datensegment, ebenfalls alle indirekten Datenzugriffe über die Register BX, SI und DI sowie ihre zulässigen Kombinationen.

Datenzugriffe über BP und die Kombinationen BP + SI und BP + DI beziehen sich dagegen auf das Stacksegment. Datenzugriffe über BP haben noch die Besonderheit, daß stets mindestens eine 1-Byte-Verschiebung vorhanden ist. Ist im Quellkode keine Verschiebung angegeben, wird vom Assembler automatisch der Wert 00 eingesetzt.

Durch ein Segment-Override-Präfix kann (meist) die standardmäßige Segment-Zuordnung aufgehoben werden. Die Angabe im Quellkode kann assemblerabhängig sein. Üblich ist etwa die Form

MOV CS: WORD PTR [BX], 0

Im Maschinenkode wird dadurch ein Vorbyte eingefügt, das für den folgenden Befehl die Segmentzuordnung ändert. Diese Vorbytes haben die hexadezimalen Werte:

ES: 26 CS: 2E SS: 36 DS: 3E

Bei den komplexen Block- oder Stringbefehlen MOVS, CMPS, SCAS, STOS und LODS sind besondere Register- und Segmentzuordnungen vorgegeben:

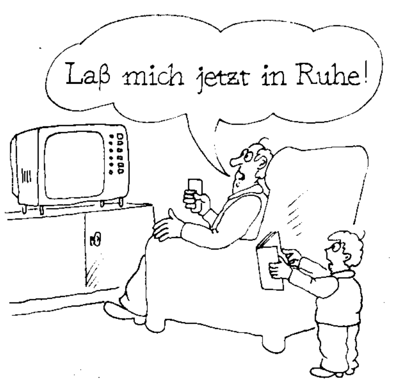
MOVS	**MOV**	**ES:[DI],**	**DS:[SI]**
CMPS	**CMP**	**ES:[DI],**	**DS:[SI]**
SCAS	**CMP**	**ES:[DI],**	**AX**
STOS	**MOV**	**ES:[DI],**	**AX**
LODS	**MOV**	**AX,**	**DS:[SI]**

Diese Aufstellung erläutert gleichzeitig die Wirkung der Stringbefehle (bis auf das automatische Inkrementieren oder Dekrementieren der Register SI und DI). Alle Stringbefehle können wort- oder bytebezogen sein. Bei den Bytebefehlen ist AX durch AL zu ersetzen. Die Verkettung des Registers DI mit dem Segmentregister ES kann bei den Stringbefehlen nicht geändert werden. Ein Segment-Override-Präfix wirkt hier nur auf die Zuordnung des Registers SI.

Dem Stackregister ist außer dem BP-Register auch der überaus wichtige Stackpointer SP zugeordnet. Da praktisch kein Programm ohne Stack auskommt (Interrupt, Befehle CALL, PUSH), muß die Kombination SS:SP stets gezielt gesetzt sein. Die Register SS und SP dürfen deshalb nur mit größter Vorsicht neu gesetzt werden. Zum Beispiel am Beginn eines neuen Programms. Maschinenprogramme ohne besondere Ansprüche an das Stackvolumen brauchen SS:SP nicht neu zu setzen und können den Stack des Betriebssystems weiter benutzen. Das hat zudem den Vorteil, daß auf Konstanten und Arbeitszellen des Betriebssystems (etwa über BP oder den Segment-Override-Präfix SS) leicht zugegriffen werden kann.

Maschinenprogramme, die einschließlich Daten- und Arbeitsbereichen mit 64 KByte Speicher auskommen, sollten zweckmäßigerweise entsprechend dem 8080-Speichermodell konzipiert werden. Dann haben zumindest die Segmentregister CS, DS und ES denselben Wert, was alle Probleme mit der Segmentregister-Zuordnung stark vereinfacht.

Dr. Walter Fürtig



Standardwortschatz

Zeichnung: Steger

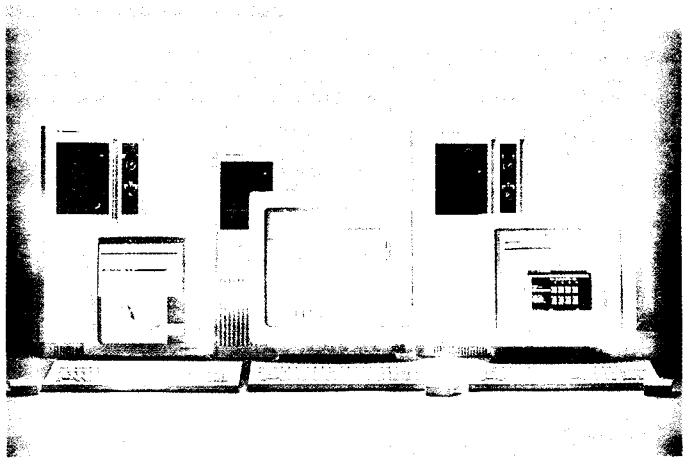
Gemeinsam für den i860

Um die Akzeptanz für Intels „Superprozessor" i860 („Eine CRAY 1 auf dem Chip.", siehe MP 8/89, S. 251) zu fördern, haben sich im September sechs Computerfirmen zum **MASS860-Konsortium** zusammengeschlossen. Alliant, IBM, OKI, Olivetti und Samsung sehen, wie die Intel Corp., die die gesamte Koordination übernimmt, als ihr Ziel die Unterstützung von Hardware-Standards und die Softwareentwicklung für auf dem 860 beruhende Systeme an. Dies soll unter anderem durch die Unterstützung eines *Multilayer Standard Application Binary Interface (ABI)* für eine große Auswahl von Hardware-Plattformen erreicht werden, um eine gute Basis an Applikations-Software für viele verschiedenartige Computersysteme zu erhalten. Das ABI ist für folgende drei Programmierumgebungen definiert:
- UNIX i860/System V Rel. 4.0 oder 860ix (ist heute für Workstation/Server-Produkte und Zusatzprozessoren verfügbar)
- APX (eine Unterkategorie von entweder UNIX-System V/i860 Rel. 4.0, OS/2 oder 860ix; wurde entwickelt, um auf PCs basierenden i860-Multiprocessing-Zusatzprozessoren zu laufen)
- PAX (erweitert den UNIX System V/i860 Rel. 4.0-Standard, um zusätzlich Hochleistungs-Multiprocessing und Parallelverarbeitungs-Systeme zu unterstützen, entwickelt von Intel und Alliant).

Die MASS860-ABI-Standards sollen es Software-Herstellern ermöglichen, eine noch nie dagewesene Auswahl von System-Plattformen mit einem einzelnen binären Release zu unterstützen. So könnte zum Beispiel eine Anwendung, welche auf einem PC-Zusatzprozessor von IBM oder Olivetti auf APX-Ebene läuft, im identischen Binärformat auf Workstations von Oki, Olivetti und Samsung wie auch auf Systemen der Supercomputing-Klasse von Alliant unter PAX laufen.

Motorola MultiPersonal Computer

Die Motorola GmbH gab kürzlich die europäische Markteinführung der neuen MultiPersonal Computer (MPC) bekannt. Mit den Systemen zielt man auf zwei wichtige Problemstellungen in den neunziger Jahren: auf das Anwenderkonzept der offenen Systeme und auf die Anwendungsmöglichkeiten von im Netzwerk betriebenen Systemen am Schreibtisch. Demzufolge berücksichtigen die MPCs eine Reihe von wichtigen Standards: das Betriebssystem Unix System V.3 (als SYSTEM V/88)), die binäre Unix-Anwenderschnittstelle ABI/BCS (Application Binary Interface/Binary Compatibility Standard), X-Open, POSIX (Portable Operating System for Computer), OSI (Open Systems Interconnection), VME-bus, X-Window-System Version 11, OSF/Motif, Ethernet-LAN-Protokoll und NSF (Network File System). Hardwaremäßig basiert das System, das in drei Modellen angeboten wird, auf dem 32-Bit-RISC-Prozessor Motorola 88100. Im Modell 100 wird er mit



20 MHz getaktet und erreicht 27 MIPS (Millionen Instruktionen je Sekunde), im Modell 200 mit 25 MHz (33,8 MIPS), und im Modell 300 kommen zwei 25-MHz-Prozessoren zum Einsatz, die bis zu 67,2 MIPS erlauben. Die Hauptspeicherkapazitäten liegen je nach Ausstattung zwischen 16 und 256 MByte, die Festplattenkapazitäten bis zu 2400 MByte. An Software sind im Lieferumfang enthalten: SoftPC, X.11, OSF/Motif, Looking Glass, FrameMaker (Demo-Version) und Uniplex.

Gleichzeitig wurde für den MPC eine Reihe neuer Netzwerk-Display-Stationen angekündigt, bestehend aus den beiden Monochrommodellen NDS 16 (16 Zoll) und NDS 19 (19 Zoll) sowie der 17-Zoll-Farbgrafikstation NDS C-17. Die X-Server sind mit den Motorola-Prozessoren MC 68000 bzw. MC 68020 und Speicherkapazitäten bis zu 4 MByte ausgerüstet. Die Software beinhaltet X-Windos, OSF/Motif und den Desktop Manager Looking Glass von Visix Software Inc.

SCSI-Controller für Drucker

Eine Konkurrenz für die bisher bei Druckern üblichen Centronics- und V.24-Schnittstellen könnte jetzt das SCSI (Small Computer Systems Interface), bisher bekannt für Festplattenbetrieb, werden. Die Firma Adaptec hat ein Controllerboard für Computer und Drucker vorgestellt, mit dem eine vom Computer generierte Bitmap ohne Umweg über eine Seitenbeschreibungssprache direkt an den Drucker geschickt und von diesem ohne Umwandlung ausgedruckt werden kann. Dabei ist auch die Datenübertragungsrate wesentlich höher. Beispielsweise dauert die Übertragung einer A4-Seite gegenüber 800 Sekunden bei V.24 und 95 Sekunden bei Centronics mit der **Image-Card 8200** von Adaptec nur noch 1,2 Sekunden.

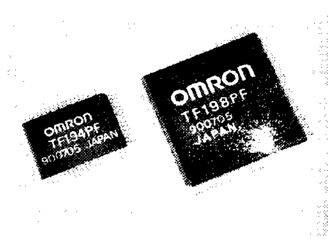
Maestro II unterstützt CSP von IBM

Mit der Software-Entwicklungsumgebung Maestro II wird Softlab im Rahmen seines AD/Cycle-Vertrages mit IBM das Cross System Product (CSP), Version 3.3, unterstützen. Maestro gilt mit 25 000 installierten Arbeitsplätzen als weltweit führende integrierte CASE-Umgebung, die sich vor allem zur Unterstützung großer Software-Entwicklungsteams eignet. CSP ist ein Werkzeug zur Anwendungsentwicklung, das die Definition von Datenstrukturen, Zugriffstechniken und prozeduralem Code gestattet und für verschiedene IBM-Zielsysteme Anwendungsprogramme generieren kann.

Die CSP-Unterstützung von Softlab wird in Form eines Datenaustausches über das externe Quellenformat von CSP geschehen. Dieses Austauschformat ist mit einfachen Dateien einsetzbar und erlaubt den Datenaustausch zum und vom CSP.

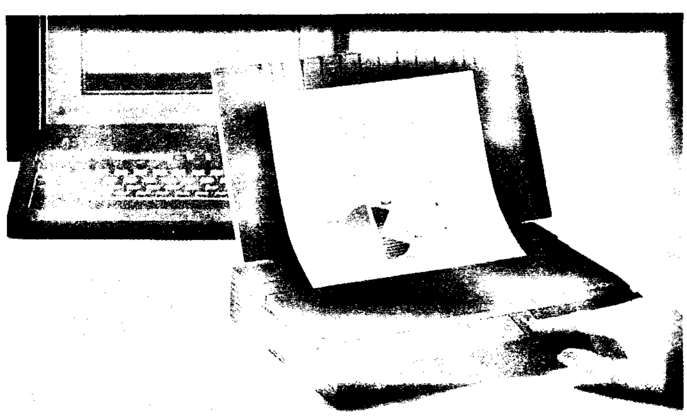
Digitale Fuzzy-Prozessoren

Die Fuzzy-Logik ist eine „unscharfe" Logik, die außer ja und nein bzw. 1 und 0 auch Aussagen bzw. Werte, die dazwischen liegen, zuläßt. Prozessoren, die diese Logik nutzen, wurden von Omron Anfang November auf der Electronica in München vorgestellt (siehe Bild). Die digitalen Fuzzy-Prozessoren (DFP) basieren auf der von Omron entwickelten Fast-Fuzzy-Prozessor-Architektur. Zwei Prozessorserien wurden bisher angeboten: die Omron-DFP-Multi-Task-Serie für umfangreiche und schnelle Prozesse wie Bild- und Informationsverarbeitung sowie die Omron-DFP-Steuerungsserie für Low-cost-Anwendungsgebiete.



Nachdem 1989 bereits der erste analoge/parallele Fuzzy-Hybrid-Schaltkreis mit 40 Regeln und einer Leistung von 8 MFLIPS (Millionen Fuzzy Logic Inference pro Sekunde) angeboten wurde, erreichen nun die neuen Prozessoren von Omron 32 000 Regeln, unterteilt in 128 Gruppen, für die parallele Verarbeitung von 128 verschiedenen Aufgaben bei einer Leistung von 10 MFLIPS.

Drucker für den Aktenkoffer



Einen neuen Drucker für unterwegs kündigte Canon an. Der BJ-10e ist ein Tintenstrahldrucker, der nach dem Bubble-Jet-Verfahren arbeitet und demzufolge Normalpapier verarbeiten kann. Herzstück ist ein 64-Düsen-Druckkopf als Bestandteil der Patroneneinheit, die zum Bedrucken von etwa 500 A4-Seiten reicht. Tintenpatrone und der Druckkopf als Verschleißteil werden also immer gemeinsam gewechselt. Die Druckgeschwindigkeit beträgt bei 10 Zeichen/Zoll 83 Zeichen/Sekunde; die Auflösung bis zu 360 dpi (dots per inch, Punkte pro Zoll). Die Stromversorgung erfolgt über Netzadapter oder Akku (als Zubehör); die Kapazität der Akkus reicht jeweils für etwa 30 Druckseiten. An den PC wird der BJ-10e über seine parallele Schnittstelle angeschlossen. Normalerweise arbeitet der Drucker im Einzelblattverfahren, jedoch läßt er sich auch mit einem automatischen Vorlageneinzug für 30 Blatt A4 ausstatten. Der BJ-10e ist 31 cm breit, 22 cm tief, 4,7 cm hoch und wiegt etwa 1,8 Kilo. Sein Preis soll unter 1000 DM liegen.

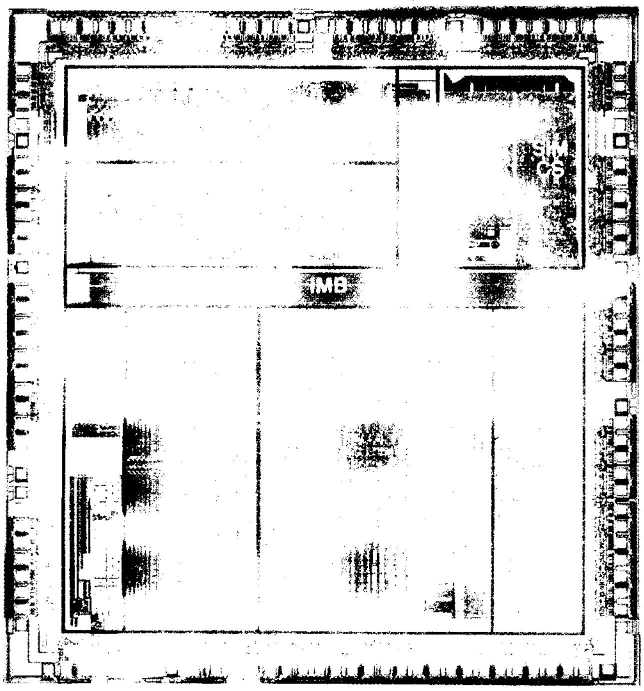
Neues von AMD

Drei Neuheiten kündigte die Advanced Micro Devices GmbH Ende September auf einer Pressekonferenz an, von denen vor allem die erste breites Interesse finden dürfte: AMD ist es gelungen, auf einem Chip neben dem Prozessor 80C286 (12/16 MHz) alle für das Funktionieren eines PC/ATs notwendigen peripheren Bausteine, die bisher auf dem Motherboard untergebracht waren, zusätzlich zu integrieren. (Bei bisher bekannten Chipsätzen mußte – u. a. aus patentrechtlichen Gründen – der Prozessorschaltkreis separat eingesetzt werden.) Es werden damit die Funktionen von bis zu 175 herkömmlichen Standardbauelementen übernommen. Beispielsweise die der AT-Standardperipherie DMA-Controller, Zähler/Zeitgeber, Interruptcontroller und Echtzeituhr sowie eines DRAM-Controllers, Buscontrollers, Peripheriecontrollers und der EMS 4.0-Speicherverwaltungslogik. Diese vom Schaltkreis **Am 286ZX** gebotenen Leistungen werden beim **AM 286LX** noch um eine spezielle Power-Managementfunktion ergänzt, die vor allem beim Einsatz in netzunabhängigen PCs Vorteile bietet. Sie beinhaltet einen CPU-Abschaltmodus, eine System-Standby-Betriebsart sowie eine verlangsamte DRAM-Refreshfunktion. Die neuen Bausteine sind vor allem für platzsparende Einstiegsmodelle bei Desktop-PCs und für tragbare PCs (Laptops und Notebook-PCs) gedacht. Zum Aufbau eines kompletten PC-ATs werden zusätzlich nur noch ein Tastaturcontroller und Speicherschaltkreise benötigt. Die Chips sollen in 216poligen PQFP-Gehäusen (28 mm^2) zum 1000er-Preis von etwa 70 Dollar/Stück angeboten werden.

Weiterhin gab die Firma bekannt, daß sie mit dem **Am 29050** ihre RISC-Prozessorfamilie 29000 (auch 29K genannt) leistungsmäßig nach oben erweitert. Im Vergleich zu den RISC-Prozessoren anderer Hersteller (Sparc, MIPS, 88000 usw.) hat der 29K zwar nicht den hohen Bekanntheitsgrad. Das mag daran liegen, daß er statt in Workstations vorrangig als Embedded Controller eingesetzt wird, seine unspektakuläre Arbeit also „eingebettet" in Druckern, Scannern, Grafikstationen, Netzwerken, X-Terminals usw. verrichtet. Hier sind jedoch jetzt schon hohe Stückzahlen zu erzielen – AMD sieht sich mit dem 29K als Marktführer –, und Prognosen zufolge werden 1993 etwa 75 Prozent aller RISC-Prozessoren auf den Embedded-Markt fallen. Beispielsweise verwendet Apple den 29K auf dem Mac-Display-Board als Grafikbeschleuniger; nach Dataquest sollen diese Anwendungen künftig nach dem Einsatz in Laserdruckern das zweitgrößte Anwendungsgebiet für 32-Bit-RISC-Prozessoren darstellen. Der 29050 bietet nun für solche Anwendungen gegenüber dem 29000 eine etwa vierfache Leistung. Er hat eine Hochleistungs-Pipeline-Gleitkommaeinheit auf dem Chip, mit der er IEEE-kompatible Arithmetik einfacher und doppelter Genauigkeit mit bis zu 80 MFLOPS (Millionen Gleitkommaoperationen je Sekunde) bei 40 MHz ausführen kann. Bei Ganzzahlverarbeitung werden 32 MIPS erreicht. Weitere Merkmale sind die Unterstützung des Burst-Mode-Zugriffs, ein Befehls-Cache mit 1024 Byte auf dem Chip (Branch Target Cache), Demand-Paging, Zeitgeber und Takterzeugung, Master-/Slave-Chip-Ausgangsüberprüfung sowie eine verbesserte Debugging-Unterstützung.

Schließlich stellte AMD noch seinen ersten Flash-Speicherschaltkreis **Am 28F010** vor. Als „Zwischending" zwischen EPROM und EEPROM haben Flashmemories vor allem den Vorteil der „blitz"schnellen elektrischen Löschbarkeit und der Möglichkeit, die Programmierung auch im System vornehmen zu können. Der Am 28F010 hat 1 MBit Kapazität (128 K × 8 Bit). Er gehört zur sogenannten ersten Generation von Flash-Memories, mit 12 Volt Programmierspannung, chipweiser Löschung und byteweiser Programmierbarkeit. Damit und in bezug auf die Anschlußbelegung ist man zum Marktführer Intel kompatibel und hofft, mit dieser Präsenz einen Industriestandard durchsetzen zu können. Abweichende Flash-Memories werden auch von Seeq, Atmel, Signetics und anderen gefertigt bzw. angekündigt. Der Am 28F010 hat von allen die kleinsten Chipabmessungen und soll mit 90 ns Zugriffszeit auch der schnellste 1-MBit-Flash-EPROM sein. Er wird im 1-μm-CMOS-Prozeß gefertigt. Mit den bereits genannten Vorteilen wird den Flash-Speichern vor allem als Ersatz oder Ergänzung von EPROMS, EEPROMS und anderen Speichermedien vor allem im Bereich der Notizbuch-PCs eine große Zukunft vorausgesagt.

Neue Controller von Motorola



Motorolas Geschäftsbereich Halbleiter präsentierte als neuestes Mitglied seiner M68300-Familie den **MC68331**, der speziell für Embedded-control-Anwendungen entwikkelt wurde. Der dem MC68332 ähnliche MC68331 enthält verschiedene Modifikationen, um eine größere Bandbreite kostensensibler Anwendungen im Leistungsbereich von 32 Bit abdecken zu können. Als Support für den MC68331 und den MC68332 bietet die Firma neben dem Hochleistungssystem CDS32 – einem leistungsstarken Entwicklungssystem für beide Prozessoren – auch das Evaluationsboard M68332EVS an. Dieses ursprünglich für den MC68332 ausgelegte Platinensystem zur Erprobung und Emulation unterstützt nun auch den MC68331. Der MC68331 ist modular aufgebaut und verfügt über eine CPU, die aus Teilen des MC68020 (CPU32) besteht. Im MC68331 sind neben der CPU32 ein Queued Serial Module (QSM), ein System Integration Module (SIM) und ein General Purpose Timer (GPT) integriert, die über den Intermodule Bus (IMB) miteinander verbunden sind. Der kleinere GPT, der die Time Processing Unit (TPU) des MC68332 ersetzt, ist zwar ebenfalls ein leistungsstarkes Zeitgeber-Modul, operiert aber nicht unabhängig von der CPU32. Der 16-Bit-Zähler des GPT, der auf einen externen oder internen Takt zugreifen kann, versorgt drei Input-capture-Kanäle, vier Output-compare-Kanäle und einen für beide Verwendungszwecke ausgelegten Kanal.

Als Ergänzung der 8-Bit-Mikrocontrollerserie MC68HC11 wurden zwei neue Bausteine angekündigt: Der **MC68HC11K4 (K4)** und der **MC68HC11711K4 (711K4)** bieten eine doppelt so hohe Busgeschwindigkeit wie die ursprünglichen Mitglieder der M68HC11-Familie und offerieren gleichzeitig eine Vielzahl von Speicherkonfigurationen sowie chipintegrierte Peripheriefunktionen. Mit ihrer höheren Busgeschwindigkeit konkurrieren die MC68HC11-Chips bereits mit vielen 16-Bit-Mikrocontrollern. Die Bausteine K4 und 711K4 besitzen einen 768 Byte großen RAM-Bereich, 640 Byte EEPROM, 62 E/A-Leitungen, einen 16-Bit-Timer, vier 8-Bit-Impulsbreiten-Modulatoren, einen achtkanaligen 8-Bit-A/D-Wandler sowie verbesserte serielle Peripherie- und Kommunikationsschnittstellen. Der K4 verfügt ergänzend über einen 24 KByte großen ROM-Bereich (Festwertspeicher), während der 711K4 mit einem lösch- und programmierbaren Festwertspeicher (EPROM) bzw. einem einmal programmierbaren 24-KByte-Festwertspeicher (OTPROM) ausgestattet ist. Beide Bausteine ermöglichen den Ausbau der Speicherkapazität bis auf 1 MByte.

Um einen leichteren Übergang von 8-Bit- auf 16-Bit-Anwendungen zu ermöglichen, bietet Motorola erstmals eine 16-Bit-Mikrocontroller-Familie an, die kompatibel zu einer Industriestandard-Produktreihe in 8-Bit-Technologie ist. Die CPU des **MC68HC16** (siehe Bild) beruht auf einer 16-Bit-Implementierung des Mikrocontrollers MC68HC11. Jeder Quellcode, der für den 8-Bit-MC68HC11 geschrieben wurde, kann unverändert für den MC68HC16 übernommen werden. Der modular aufgebaute MC68HC16 verwendet bereits vorhandene Module der Motorola-Produktfamilie M68300. Der MC68HC16 ist auch – so Motorola – der erste General-purpose-Mikrocontroller, der digitale Signalverarbeitungsfunktionen für den Einsatz in Steuerungssystemen enthält.

Der in 1-μm-HCMOS-Technologie gefertigte MC68HC16Z1 (HC16Z1) ist das erste Produkt der neuen Reihe. Die CPU ist von einer großen Zahl von Peripheriefunktionen umgeben, die modular in den Chip integriert sind. Die modulare Architektur und der Intermodule Bus (IMB) des HC16Z1 wurden aus der 32-Bit-Familie M68300 übernommen. Auch andere Module des HC16Z1 profitieren von Entwicklungen, die ursprünglich für andere M68300-Bausteine getätigt wurden. Beispiele hierfür sind das QSM und das SIM. Die CPU des MC68HC16Z1 unterscheidet sich von der CPU des MC68HC11 durch drei zusätzliche Multiplizieranweisungen, zwei neue Divisionsbefehle, neue Adressierungsarten und zusätzliche Register. Die eingebauten Digital-Signal-Processing-Funktionen gestatten in einigen Anwendungen den Verzicht auf externen DSP-Support.

Novell NetWare für den Macintosh

Seit September wird von Novell eine neue Version von NetWare für den Macintosh angeboten. Mit dem neuen Release 2.0 soll Macintosh-Anwendern ein problemloser Zugang zu allen Diensten und Dateien unter NetWare 2.15 zur Verfügung gestellt werden und die volle Integration der Apple-Rechner in die Net-Ware-Umgebung einschließlich der Verbindungen zu MS-DOS und OS/2-Anwendern garantieren. NetWare für den Macintosh 2.0 setzt auf dem Netzwerkbetriebssystem NetWare 2.15 von Novell auf und beinhaltet sowohl alle NetWare-Funktionen als auch die des AppleTalk Filing Protocols 2.0 (AFP). Anwender, die NetWare 2.15 nach dem 1. August 1990 gekauft haben, erhalten auf Wunsch ein kostenloses Update von NetWare für den Macintosh 2.0.

Multiplan in der Praxis

von J. Bernert und M. Burow, Verlag Technik, Berlin 1990, 240 S., 29,80 DM

Mit ihrem Buch wenden sich die Autoren sowohl an Anfänger als auch an Fortgeschrittene in der Tabellenkalkulation. Das war unter dem Blickwinkel des eingeschränkten Buchangebotes auf dem Gebiet der ehemaligen DDR verständlich. Heute wäre es legitim, für jede dieser Zielgruppen ein eigenes Buch herauszubringen. Trotzdem bietet das Werk für beide Zielgruppen Wesentliches.

Für den Anfänger sind ausführliche Erläuterungen zum Gegenstand zu den Termini der Tabellenkalkulation am Anfang gedacht. Die Beschreibungen der einzelnen Befehle, Funktionen und des Makrokonzepts werden mit verbalen Erklärungen eingeleitet. Ein Abschnitt mit Beispielen unterstützt die Einarbeitung in Multiplan. Wünschenswert für diese Zielgruppe wären auch umfangreichere Beispiele in den vorderen Abschnitten gewesen. Das Buch ist kein eigentliches Lehrbuch, mit den enthaltenen Hinweisen zum Lesen und bei Beachtung der zahlreichen Querverweise kann sich ein Anfänger aber in die Materie einarbeiten.

Der fortgeschrittene Leser begrüßt die handbuchartige, aber trotzdem umfassende und sehr sorgfältige Vorstellung der Sprachelemente und Systemfunktionen. Für beide Zielgruppen hilfreich sind die vielen praktischen Hinweise und Tips, die Klippen umgehen und den Einsatz von Multiplan rationalisieren helfen. Das betrifft auch die Bemerkungen zu Fehlermeldungen des Systems, zu den Unterschieden der Versionen und zum Einsatz auf unterschiedlicher Hardware. *R. Mundstock*

Wissensspeicher 80286-Programmierung

von L. Claßen und U. Wiesner, Verlag Technik, Berlin 1990, 239 S., 24,– DM

Keine Frage: Als Herzstück des Standard-PCs unserer Tage kommt dem 80286 besondere Bedeutung zu. Das Buch bietet Ihnen dazu, was man von einem guten Wissensspeicher erwarten darf: umfassende und schnell auffindbare Information. Es erspart dem Anfänger wohl nicht die einführende Literatur, mag aber für den aufsteigenden 8086-Programmierer bereits ausreichend sein. Für ihn ist auch der Abschnitt zur Kompatibilität beider Prozessoren von besonderem Interesse, der die Unterschiede zusammenfaßt und Hinweise zum Schreiben portierbaren Codes gibt. Daß die Schaffung eines Nachschlagewerkes primäres Ziel war, zeigt die Redundanz: Der Befehlssatz wird in einer ausführlichen Beschreibung (etwa 100 Seiten), als übersichtliche Befehlsliste im Anhang und als Tabelle der Opcodes dargestellt.

Die Betriebsart Protected Mode des Prozessors – eine neue Qualität des 80286 gegenüber dem 8086 – bringt einen größeren (virtuellen) Adreßraum und – bis dato vernachlässigte – Zugriffskonzepte. Moderne Betriebssysteme fordern vier Privilegierungsstufen, wie sie der 80286 bietet, auch wenn häufig nur ein zweistufiges Konzept realisiert wird. In engem Zusammenhang damit steht die Erweiterung des Interruptsystems zur Behandlung von Verletzungen der Zugriffsrechte. Der Speicherverwaltung, dem Zugriffsschutz und dem Interruptsystem ist je ein Kapitel gewidmet. Hinzu kommen die Ein- und Ausgabe sowie das neuartige Taskkonzept im Protected Mode des Prozessors, womit Multitasking direkt unterstützt wird.

Bei aller Kürze ist die Darstellung verständlich und mit einer Reihe von Abbildungen illustriert. Wer sich an die Programmierung im Protected Mode wagt, wird zudem wissen, worauf er sich einläßt, so daß auch Worte wie Lokaldeskriptortabellenregister leicht von der Zunge gehen sollten.

Da sich die Darstellung nur auf den Prozessor bezieht, finden Sie keine Hinweise zur Arbeit mit MASM (oder TASM). Der Anhang liefert aber ein mehrseitiges Assemblerprogramm zur Initialisierung des Prozessors im Protected Mode unter Ausnutzung der Schutzmechanismen.

Bei aller Freude über die Fortführung der – ehemals so wertvollen – Reihe Technische Informatik würde der Leser auch etwas weniger Traditionsbewußtsein (sprich: Druck nach Wordstar-Layout des Autors) verschmerzen. (Bessere Papierqualität wage ich erst gar nicht zu fordern.) Zum angegebenen Preis ist das Buch aber in jedem Fall eine Überlegung wert. Auch auf dem großen und freien Markt sind Bücher mit kompakten, aber verständlich aufbereiteten Informationen nicht unbedingt dicht gesät. *U. Schulze*

Der Verlagskaufmann

von R. Mundhenke, 5. erweiterte Auflage, Societäts-Verlag, Frankfurt/Main 1988, 720 S., ISBN 3-7973-0431-5, 56,– DM

Hat vielleicht auch Sie das Gründungsfieber gepackt? In unseren Heften 9 und 10/90 hatten wir Ihnen ja bereits einige Hinweise gegeben, wie Sie ihren Traum von der Selbständigkeit leichter verwirklichen können. Sollten Sie eine Zukunft im Verlagsgewerbe ins Auge gefaßt haben, könnte Ihnen das hier vorgestellte Buch mit Sicherheit wertvolle Hilfestellung bieten. Im Zeitalter des Desktop Publishing glaubt ja so mancher, daß der Schritt vom gelegentlichen Herstellen von Druckschriften zum professionellen Publizieren nicht so groß sein kann. Mit diesem Buch erhalten Sie nicht nur einen Überblick über den Aufbau eines Verlages, sondern auch Hinweise über die in den einzelnen Bereichen notwendigen Aktivitäten. Untergliedert wird das umfangreiche Werk in die Kapitel Verlagsgewerbe, Vertriebsorganisation, Anzeigengeschäft, Werbung, Rechnungswesen, Redaktion und Lektorat, Drucktechnische Herstellung und Die neuen Medien. Bemerkenswert sind ein umfangreiches Fachwörterlexikon sowie Verzeichnisse von Fachbüchern und Fachzeitschriften des Verlagswesens sowie nützliche Anschriften von Fachverbänden und anderen Institutionen.

Und da das Buch ja eine Berufsfachkunde – also ein Lehrbuch für Auszubildende und Ausbilder – ist, fehlen auch nicht Aufgaben zum verlagskaufmännischen Rechnen sowie programmierte Textaufgaben zum Üben. In dieser 5. überarbeiteten Auflage wurden gegenüber den bisherigen zahlreiche Veränderungen vorgenommen. Beispielsweise beim ZAW-Rahmenschema für Werbeträgeranalysen oder mit der Aufnahme von Vertriebsrichtlinien.

Vermutlich unter dem Eindruck der einige Zeit zurückliegenden Euphorie über Videotext und Bildschirmtext hat der Autor diesen das genannte Kapitel „Die neuen Medien" gewidmet. Aus heutiger Sicht wäre es sicher notwendiger, andere Fragen des Computereinsatzes im Verlag ausführlich zu behandeln – angefangen von der Textverarbeitung über Archivsysteme und Finanzbuchhaltung bis zum Einsatz von Satzsystemen. Denn ob sich ein Verlag für den Satz – vielleicht gar die Belichtung – im Haus entscheidet oder diese weiter als Lohnarbeit ausführen läßt, beeinflußt heutzutage ganz erheblich seine Wettbewerbs- und damit Existenzfähigkeit. Hier könnten Hinweise zu Strukturveränderungen aus kaufmännischer Sicht sehr hilfreich sein.

Wünschenswert wäre also eine weitere Überarbeitung, die den Auswirkungen, die „Kollege Computer" verursacht, weitaus mehr Raum widmet. Und der „Kollege" läßt uns nicht viel Zeit! *H. Weiß*

Turbo-Assembler

Das Buch für den Einstieg

von P. Monadjemi, Addison-Wesley (Deutschland) GmbH, Bonn 1990, 587 S., mit Programmdiskette, 68,– DM, ISBN: 3-89319-225-5

Ein Buch zum Turbo-Assembler muß sich insbesondere daran messen lassen, was es mehr bietet als die gewohnt guten deutschen Handbücher, die jeder Borland-Kunde bei Erwerb der Software erhält. In diesem Fall soll es die Einführung in die Assemblerprogrammierung sein. Daß dafür ein konkretes Produkt gewählt wurde, ist eher ein Vorteil, denn der Teufel steckt – und das gilt für einen Assembler besonders – im Detail.

Obwohl der Turbo-Assembler gegenüber seinem Microsoft-Pendant mit einer Reihe von Verbesserungen aufwarten kann (das Erzwingen einer strengeren Notation würde ich auch zu den Vorteilen zählen), orientiert sich das Buch weitgehend am Defacto-Standard des Marktführers. Hier gefällt die Ehrlichkeit des Autors, der klarmacht, daß an MASM kaum ein Weg vorbeiführt. Für den angehenden Programmierer hat das den Vorteil, daß er – sofern das Kapitel fortgeschrittener TASM-Programmierung überlesen wird – auch leicht mit einem anderen Assembler zurechtkommt.

Der Leser wird nicht lange mit der Vorrede aufgehalten. Nach ein paar einführenden Bemerkungen („Ist Assemblerprogrammierung noch in?") heißt es: Rechner einschalten und die ersten Beispiele ausprobieren. Begonnen wird mit Installation und Aufruf von Assembler und Linker. Ein wenig Theorie bleibt Ihnen nicht erspart: Es geht um den Stack, binäre Logik und Interrupts sowie die (reichlich umständliche) Adressierung des 8086. Die folgenden Kapitel befassen sich mit dem allgemeinen Aufbau eines Assemblerprogramms (vor allem Segmentierung und Speichermodelle), dem Befehlssatz und Möglichkeiten der Strukturierung. Ein eigener Abschnitt ist der Nutzung der DOS-Schnittstelle gewidmet, ohne die kein Assemblerprogramm auskommt.

Die Beispiele sind vielseitig verwendbar, man sollte sich wirklich die Mühe machen, sie auszuprobieren. Nur so wird man sich bei späteren Arbeiten daran erinnern und darauf zurückgreifen.

Das Buch ist dick genug, auch fortgeschrittener Assemblerprogrammierung Platz einzuräumen. Dazu gehören Macros, die Verbindung mehrerer Module und die Hochspracheneinbindung. Letzteres ist von besonderer Bedeutung – wenden sich doch viele Programmierer einem Assembler mit dem Ziel zu, ihre Hochsprachenprogramme zu tunen. Parallel zur Programmentwicklung beschreibt der Autor auch die Arbeit mit dem Turbo-Debugger – wohl wissend, daß die Fehlersuche häufig die meiste Zeit in Anspruch nimmt.

Es sollte dem Neuling keine Schwierigkeiten bereiten, das Buch durchzuarbeiten. Der Schreibstil ist flüssig, die Erläuterungen anschaulich. Die gute Druckqualität trägt ebenso zur leichten Lesbarkeit bei, wie der sinnvolle Einsatz von (zuweilen recht witzigen) Fußnoten. (Fußnote zum Adreßraum der CPU: „Bei den heutigen Preisen für dynamische RAMs (DRAMs) wäre ein voll ausgebauter Arbeitsspeicher eine interessante Alternative zum Einfamilienhaus".)

Weniger Freude bereitet das Nachschlagen. Die Zusammenfassung der Befehle im Anhang ist etwas unübersichtlich und die (sehr wohl beschriebenen) Direktiven MASM und IDEAL sucht man im TASM-Index vergeblich.

Fazit: Wer schon einmal Assembler programmiert hat oder von MASM umsteigen möchte, ist mit den Borland-Handbüchern wohl ausreichend bedient. Wollen Sie sich aber erstmals an einem Assembler versuchen und haben dafür den Turbo-Assembler ausgewählt, so erhalten Sie hier auf fast 600 Seiten und einer Diskette Hilfe zu einem fairen Preis. *U. Schulze*

Fortsetzung von der 2. Umschlagseite

zu montierenden Fotos wurden die Flächen bereits freigestellt. Bei der Vorbereitung des Beitrages zeigten sich auch die Grenzen von WordPerfect. Um die äußere Form der MP zu erreichen, mußten einige Tricks angewendet werden. Ein DTP-Programm, wie der Ventura-Publisher, unterstützt solche Arbeiten wesentlich besser.

Wie bei Textverarbeitungsprogrammen inzwischen allgemein üblich, verfügt auch WordPerfect über die Möglichkeit der Druckvorschau (Page Preview) auf dem Bildschirm. Gegenüberliegende Seiten können gleichzeitig eingesehen werden, drei verschiedene Größen stehen zur Auswahl. Auf diese Weise läßt sich einiges an Papier sparen. Gut unterstützt wird die Einbindung von Grafiken der gängigen Formate, etwa von gescannten oder mit Zeichenprogrammen erzeugten Bildern (Bild 2). WordPercfect verwendet ein eigenes Grafikformat WPG, unterstützt aber 13 gängige Formate anderer Programme, darunter solche Standards wie TIFF, HPGL oder CGM. Damit lassen sich Bilder aus den meisten populären Zeichen-, Mal - oder Konstruktionsprogrammen einbinden, von denen nur einige genannt werden sollen: AutoCad, Dr. Halo, GEM Draw und Paint, Harvard Graphics, Pizazz und Windows Draw und Paint.

Da WordPerfect im Textmodus arbeitet, wird für die Grafiken nur ein Rahmen freigehalten, in den das entsprechende Bild erst beim Druck eingefügt wird. Es ist möglich, Grafiken zu drehen, zu spiegeln und zu invertieren (bei Vektorgrafik), sowie zu vergrößern und zu verkleinern (Pixelgrafik). Das Zeichnen von Rahmen mit dem erweiterten ASCII-Zeichensatz kann direkt über die Cursortasten erfolgen - so spart man sich die komplizierte Eingabe von Steuerkodes mit der Alt-Taste.

Bei der Vorstellung der Version 5.1 wurde vom Hersteller besonders auf den neuen Formeleditor hingewiesen (vgl. MP 8/90, Seite 40). Formeln werden eingegeben wie gesprochen, zum Beispiel:

{a times b} over c

Ein Menü ermöglicht aber auch ein Erstellen im Dialog. Die Darstellung erfolgt grafisch, das heißt, es wird eine Grafikbox reserviert. Um die Darstellung einzusehen, braucht aber nicht die Preview-Funktion bemüht zu werden; der Formeleditor zeigt das Ergebnis in einem Fenster ständig an. Der Formeleditor kennt alle wichtigen Operatoren und eine Vielzahl von Sonderzeichen (z.B. das ganze griechische Alphabet). Wenngleich für diese Funktionen nur ein kleiner Anwenderkreis in Frage kommt, so sucht sie doch ihresgleichen bei anderen Textverarbeitungsprogrammen. Von größerem Interesse dürfte die erweiterte Funktion zur Tabellenkalkulation sein, die einem Spreadsheetprogramm ähnelt. Eine Tabelle wird aus den angegebenen Zeilen und Spalten generiert (Bild 3), und die Felder können mit Hilfe von Formeln verknüpft werden, um eine automatische Neuberechnung von abhängigen Feldern zu gewährleisten.

Außerdem existiert eine Schnittstelle für den Import von Tabellen aus Excel, Lotus und PlanPerfect. Sie schließt eine echte Kopplung ein, so daß Änderungen in der Tabelle automatisch in den WordPerfect-Text übernommen werden.

Die verschiedenen Versionen von WordPerfect unterscheiden sich zwar in ihrem internen Format, werden aber beim Laden automatisch konvertiert. Interessant ist in diesem Zusammenhang, daß die Version 5.0 Texte, die mit 5.1 erstellt wurden, als *neuere Version* erkennt und auch korrekt umwandelt. Beim Laden von Wordstar-Dateien wird ebenfalls automatisch das WordPerfect-Format erzeugt. Sie werden das zu schätzen wissen, wenn Sie sich schon einmal mit den verschiedenen Wordstar-Versionen befassen mußten. Obwohl das mitgelieferte Programm Convert das Hausformat in alle gängigen Textformate überführt, enthält WordPerfect zusätzlich eine leistungsfähige Makrosprache, mit der Texte nachträglich bearbeitet oder konvertiert werden können. Besonders die Erzeugung stets gleicher Formate läßt sich damit gut automatisieren.

Bei Benutzung der Rechtschreibprüfung ist mir aufgefallen, daß oftmals einige Verbformen fehlen. So werden vom Verb *laufen* die Partizipien *laufend* und *gelaufen* gefunden, ebenso *ich laufe* und *er läuft* - nicht aber *ihr lauft* und *du läufst.*

Zu den Kleinigkeiten, die man schnell schätzen lernt, gehört die Auto-Save-Funktion. Nach Ablauf eines festgelegten Zeitintervalls wird der Text stets automatisch auf der Platte gesichert.

Ein großes Plus von WordPerfect gegenüber den Konkurrenten Wordstar und MS-Word besteht in der Verfügbarkeit auch für andere Betriebssysteme. So kann die Version 4.2 für VMS und einige Unix-Derivate bezogen werden. Word-Perfect wird sowohl auf elf 5 ¼ Zoll-, als auch sechs 3 ½ Zoll-Disketten geliefert (360 bzw. 720 KByte). Weiterhin erhalten Sie drei dicke Handbücher (1 x Einführung, 2 x Referenz) und die erwähnte Tastaturschablone. Ein Hauptspeicher von 384 KByte ist ausreichend.

Um Ihnen die Entscheidung eines eventuellen Umstiegs auf die Version 5.1 zu erleichtern, hier noch einmal die Neuheiten zusammengefaßt:

- Pull-down-Menüs
- Mausunterstützung
- Formeleditor
- verbesserte Tabellenfunktion
- kontextsensitives Hilfssystem
- erweiterter Font von Sonderzeichen
- Verwaltung erweiterter Dateinamen und Zusatzinformationen.

Insgesamt wurde WordPerfect gegenüber der Version 5.0 um 30 Funktionen erweitert, die vertraute Umgebung blieb aber erhalten.

Uwe Schulze

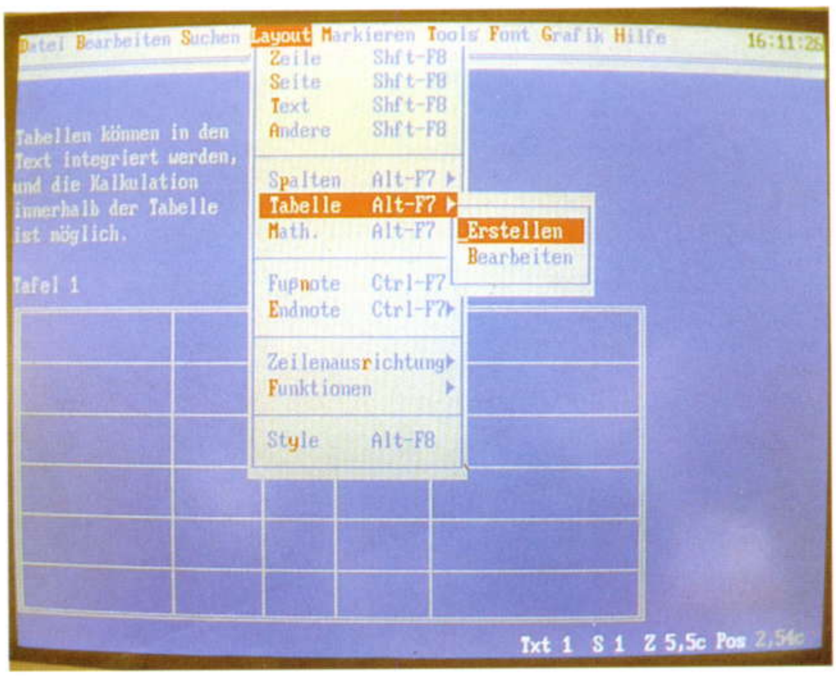


Bild 3

Auslieferung in diesen Tagen

Programmierhandbuch 80386/80486

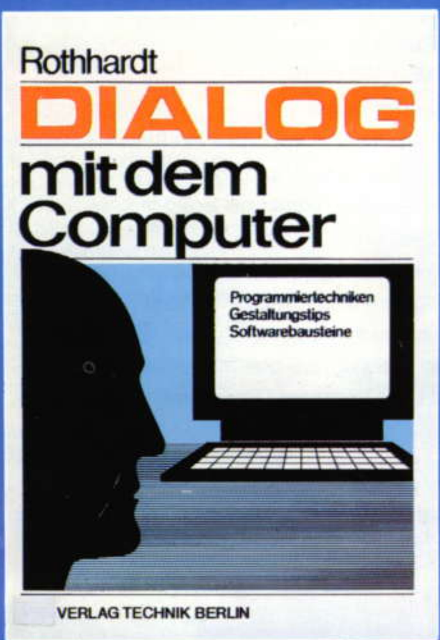
Von Dr.-Ing. Ludwig Claßen, 1. Auflage. 240 Seiten, 73 Bilder, 24 Tafeln, Broschur, 29,80 DM, ISBN 3-341-01033-5

Die Mikroprozessoren 80386 und 80486 werden in unzähligen MS-DOS-, OS/2- und Unix-PCs eingesetzt. Das Buch behandelt die Softwarearchitektur der beiden 32-Bit-Prozessoren. Bei der Einarbeitung in die Softwarearchitektur der Mikroprozessoren 80386 und 80486 werden Grundkenntnisse der Programmierung vorausgesetzt. Für den erfahrenen Programmierer hat der Band den Charakter eines Nachschlagewerkes mit hoher Informationsdichte.

Clipper mit Version 5.0

Von Dipl.-Ing. Uwe Holz und Dr.-Ing. Lutz-Ulrich Plogas. 1. Auflage. 240 Seiten, 11 Bilder, 26 Tafeln, Broschur, 29,80 DM, ISBN 3-341-01031-9

Der 1984 als erster dBase-kompatibler Compiler angebotene Clipper Winter '84 wurde schnell zu einem eigenständigen Entwicklungssystem für PC-Datenbanken. Das vorliegende Buch ist in drei Teile geteilt und somit sowohl für Anfänger als auch für Fortgeschrittene und Profis ein einführendes Lehrbuch, ein informatives Handbuch und ein anspruchsvolles Fachbuch in einem. Es konzentriert sich darauf, dem Anwender Hinweise und Anregungen für die praktische Arbeit mit Clipper zu geben und geht somit weit über eine allgemeine Sprachbeschreibung hinaus.



Dialog mit dem Computer

Programmiertechniken · Gestaltungstips · Softwarebausteine

Von Dr.-Ing. Günter Rothhardt. 1. Auflage. 264 Seiten, 51 Bilder, 13 Tafeln, 9 Programme, Leinen, 34,– DM. ISBN 3-341-00860-8

Elementare Programmierkenntnisse sind ausreichend, um die in diesem Buch enthaltenen benutzerfreundlichen Dialoganwendungen für Heim- und Personalcomputer verstehen zu können. Der Inhalt umfaßt die Gestaltungsregeln für alle gebräuchlichen Dialogtechniken, anschauliche Programmierbeispiele und Anregungen für eigene Anwendungen.

MULTIPLAN in der Praxis

Von Ing. Joachim Bernert und Dipl.-Ök. Michael Burow. 1. Auflage. 240 Seiten, 66 Bilder, 28 Tafeln, Broschur, 29,80 DM. ISBN 3-341-01011-4

Der Anfänger findet hier das notwendige Einstiegswissen für die Nutzung von Tabellenkalkulationsprogrammen. Der fortgeschrittene Anwender erhält eine Vielzahl praktischer Hinweise unter anderem zur Übernahme von Tabellen aus älteren Multiplanversionen, zum Datenübergang in das Geschäftsgrafikprogramm MS-Chart und zum Programm Sideways.

Bestellschein

Ich/Wir bestellen aus dem Verlag Technik Berlin

__ Expl. 3-341-01033-5/Claßen, 80386/80486	29,80 DM
__ Expl. 3-341-01031-9/Holz/Plogas, Clipper	29,80 DM
__ Expl. 3-341-00860-8/Rothardt, Dialog	34,00 DM
__ Expl. 3-341-01011-4/Bernert, MULTIPLAN	etwa 29,80 DM

Den Bestellschein senden Sie bitte an Ihre Buchhandlung oder direkt an die Verlag Technik GmbH, Postfach 201, O–1020 Berlin

Info-Service

Ich/Wir wünschen kostenlos und unverbindlich Informationsmaterial über Neuerscheinungen aus dem Verlag Technik auf folgenden Gebieten:

- ☐ Maschinenbau
- ☐ Wärme-, Luft- u. Kältetechnik
- ☐ Feinmechanik/Optik
- ☐ Elektrotechnik
- ☐ Elektronik
- ☐ Meßtechnik
- ☐ Automatisierungstechnik
- ☐ Informatik
- ☐ Physik
- ☐ Wörterbücher für Wissenschaft und Technik
- ☐ Populärwissenschaftliche Bücher
- ☐ Fachzeitschriften

Bitte teilen Sie uns Gewünschtes mit.